MATHEMATICS OF DEEP LEARNING

A Self-Contained Modern Treatise

PART I

THE GEOMETRIC FOUNDATIONS

Linear Algebra and SVD • Backpropagation as Automatic Differentiation
Information Geometry and the Fisher Metric • Natural Gradient and K-FAC

Eight-Part Monograph Series on the Mathematical Principles of Neural Networks
Independent of Companion Volumes — Complete in Itself

Preface: On the architecture of this treatise

This monograph is the first of eight self-contained parts comprising a complete mathematical treatment of deep learning. Unlike companion works in this series, it is designed to be read in isolation: every prerequisite concept is introduced from first principles, every major theorem is stated precisely and proved to the degree that conveys the essential mechanism, and every technical detour is justified by its payoff in later chapters. The reader who completes this part will possess the geometric vocabulary necessary to understand not merely how neural networks are trained, but why the mathematical structures chosen by practitioners are the ones that work.

The four pillars of this volume are chosen deliberately. Linear algebra and the singular value decomposition provide the static geometry of weight matrices and representational spaces. Backpropagation, reframed as automatic differentiation on computation graphs, provides the dynamic machinery by which gradients propagate through this geometry. Information geometry introduces the curvature of the space of probability distributions, revealing why naive gradient descent on neural networks is poorly conditioned. Natural gradient methods and their tractable approximation via K-FAC close the circle, translating the theoretical insight of information geometry into practical second-order optimisers that represent the current frontier of training efficiency.

Throughout, we maintain a discipline of precision without pedantry. Every definition is motivated before it is stated; every theorem is accompanied by a discussion of its significance; every formula is parsed in natural language before being manipulated algebraically. The reader we have in mind is a graduate student or researcher comfortable with multivariable calculus and basic linear algebra who seeks a rigorous but navigable account of the mathematical depth underlying modern neural network training.

On Notation

We adopt the following conventions throughout. Scalars are italic lowercase: a, b, λ. Vectors are bold lowercase: $\boldsymbol{\theta}$, $\mathbf{x}$, $\mathbf{g}$. Matrices are uppercase Roman: $\mathbf{W}$, $\mathbf{A}$, $\mathbf{G}$. Tensors of order three or higher are denoted with calligraphic capitals: $\mathcal{T}$, $\mathcal{H}$. The symbol ∇ denotes a gradient with respect to its subscripted argument; ∂ denotes a partial derivative. The notation $\nabla_{\boldsymbol{\theta}} L$ abbreviates $\partial L / \partial \boldsymbol{\theta}$. Expectations are written $\mathbb{E}_{p}[\cdot]$ with subscript indicating the probability measure. The Frobenius norm of a matrix is $\|\mathbf{A}\|_{F}$; the spectral norm (largest singular value) is $\|\mathbf{A}\|_{2}$; the nuclear norm (sum of singular values) is $\|\mathbf{A}\|_{*}$. We write $\mathbf{A} \succeq 0$ to denote positive semidefiniteness and $\mathbf{A} \succ 0$ for positive definiteness. The Kronecker product is $\mathbf{A} \otimes \mathbf{B}$; the Hadamard (elementwise) product is $\mathbf{A} \odot \mathbf{B}$. The symbol $\triangleq$ denotes definitional equality; $\approx$ denotes numerical approximation; $\sim$ denotes distributional equality. Log always means natural logarithm unless otherwise specified.

Chapter 1: Linear Algebra and the Singular Value Decomposition

Every computation a neural network performs is, at its most fundamental level, a sequence of linear maps interleaved with pointwise nonlinearities. This observation, which sounds trivially reductive, is in fact mathematically rich. The linear maps — encoded as weight matrices — carry the learned structure of a model; their geometry determines what features are represented, how information flows forward, and how errors propagate backward. Understanding the singular value decomposition is therefore not merely an exercise in matrix algebra: it is the foundation of the geometric intuition required to reason about neural networks in depth.

1.1 Vector Spaces, Linear Maps, and the Geometry of Representations

A vector space V over $\mathbb{R}$ is a set closed under vector addition and scalar multiplication satisfying the usual axioms of commutativity, associativity, distributivity, and the existence of additive identity and inverses. The dimension of V, denoted $\dim(V)$, is the cardinality of any basis — a maximal linearly independent spanning set. In deep learning, V is almost always $\mathbb{R}^n$ for some integer n, and we think of elements of V as activation vectors: the output of one layer passed as input to the next.

A linear map $T: V \to W$ between finite-dimensional vector spaces is completely determined by its action on a basis. If $\{e_1, \ldots, e_n\}$ is a basis for V and $\{f_1, \ldots, f_m\}$ is a basis for W, then T is encoded by the matrix $W \in \mathbb{R}^{m \times n}$ with columns $W_j = T(e_j)$. Composing two linear maps corresponds to matrix multiplication; the adjoint of T corresponds to the transpose W^T. This correspondence between linear maps and matrices is the reason all of deep learning can be reduced to matrix algebra.

Fundamental subspaces. Every matrix $W \in \mathbb{R}^{m \times n}$ induces four fundamental subspaces that partition $\mathbb{R}^n$ and $\mathbb{R}^m$: the column space $\text{Col}(W) \subset \mathbb{R}^m$, the null space $\text{Null}(W) \subset \mathbb{R}^n$, the row space $\text{Row}(W) = \text{Col}(W^T) \subset \mathbb{R}^n$, and the left null space $\text{Null}(W^T) \subset \mathbb{R}^m$. The rank-nullity theorem establishes that $\dim(\text{Col}(W)) + \dim(\text{Null}(W)) = n$, so the rank $r = \dim(\text{Col}(W))$ and the nullity $n - r$ partition the input space. For neural networks, the null space of a weight matrix represents directions in the input that are completely suppressed — irretrievable information that cannot affect the output regardless of what comes after in the network.

The rank of a weight matrix is a fundamental measure of its expressivity. A full-rank matrix $W \in \mathbb{R}^{m \times n}$ with $r = \min(m,n)$ represents an injective (if $m \geq n$) or surjective (if $m \leq n$) linear map; no information is lost in the former case, and the map spans its entire codomain in the latter. In practice, trained weight matrices are rarely full rank, and the degree of rank deficiency encodes the extent to which the network has specialised its representations. This observation motivates the singular value decomposition.

1.2 The Singular Value Decomposition: Existence, Uniqueness, and Geometry

The singular value decomposition (SVD) is the most important factorisation in numerical linear algebra and, by extension, in the mathematical analysis of neural networks. Its statement is deceptively simple, but each component carries deep geometric meaning.

Theorem 1.1 (Singular Value Decomposition — Existence and Uniqueness)

Let $W \in \mathbb{R}^{m \times n}$ with rank r. There exist orthogonal matrices $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ and a matrix $\Sigma \in \mathbb{R}^{m \times n}$ with nonnegative entries $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0 = \sigma_{r+1} = \ldots$ on its main diagonal and zeros elsewhere, such that:

$$W = U \Sigma V^T$$

The values σ_i are unique and called the singular values of W. The columns u_i of U (left singular vectors) and v_i of V (right singular vectors) are unique up to simultaneous sign flips in matched pairs and, for repeated singular values, up to orthogonal rotation within the corresponding eigenspaces of

WW^T and W^TW respectively. The singular values satisfy $\sigma_i^2 = \lambda_i(WW^T) = \lambda_i(W^TW)$, linking SVD to the eigendecomposition of the associated symmetric matrices.

Geometric interpretation. The SVD decomposes the action of W into three successive operations: (1) V^T rotates the input into the coordinate system aligned with the principal directions of W; (2) Σ scales each principal direction by the corresponding singular value, stretching or compressing the input in a diagonal fashion; (3) U rotates the scaled input into the output space. The image of the unit ball under W is therefore an ellipsoid with semi-axis lengths $\sigma_1, ..., \sigma_r$ aligned along the directions $u_1, ..., u_r$.

The thin SVD retains only the r nonzero singular values, writing $W = U_r \Sigma_r V_r^T$ where $U_r \in \mathbb{R}^{m \times r}$ and $V_r \in \mathbb{R}^{n \times r}$. This is the computationally relevant form for matrices of low rank. The outer product expansion $W = \Sigma_i \sigma_i u_i v_i^T$ decomposes W as a sum of rank-one matrices, each of magnitude σ_i. Truncating this sum at $k < r$ terms yields the best rank-k approximation to W, which is the content of the Eckart-Young-Mirsky theorem.

Theorem 1.2 (Eckart-Young-Mirsky — Best Low-Rank Approximation)

Let $W = U\Sigma V^T$ be the SVD of $W \in \mathbb{R}^{m \times n}$, and let $W_k = \Sigma_{i=1}^{k} \sigma_i u_i v_i^T$ be the rank-k truncation. Then:

$\min_{r: \text{rank}(B) \leq k} \|W - B\|_t = \|W - W_k\|_t = \sigma_{k+1}$ (Frobenius norm)

$\min_{r: \text{rank}(B) \leq k} \|W - B\|_2 = \|W - W_k\|_2 = \sigma_{k+1}$ (spectral norm)

The same truncated SVD is optimal simultaneously for both norms. This remarkable fact follows from the unitary invariance of both norms and the interlacing properties of singular values. For any unitarily invariant norm, W_k is the unique best rank-k approximation when $\sigma_k > \sigma_{k+1}$ (strict inequality).

The practical significance of Eckart-Young-Mirsky for deep learning is profound. First, it provides the theoretical basis for LoRA (Low-Rank Adaptation) and related parameter-efficient fine-tuning methods: if the relevant update ΔW to a pre-trained weight matrix W_0 lies in a low-

dimensional subspace, then a low-rank approximation $\Delta W \approx BA$ with $B \in \mathbb{R}^{m \times r}$ and $A \in \mathbb{R}^{r \times n}$ can represent it faithfully with a dramatic reduction in parameter count. Second, it motivates spectral pruning: singular values below a threshold ε can be zeroed without loss of approximation quality greater than ε in spectral norm.

1.3 The Spectral Theorem and Positive Definite Matrices

Symmetric positive definite (SPD) matrices appear throughout the mathematics of neural networks: the Fisher information matrix, the Hessian of the loss, the Gram matrix of a kernel. Their algebraic properties are considerably richer than those of general matrices, and the Spectral Theorem provides the bridge between algebraic and geometric intuitions.

Theorem 1.3 (Spectral Theorem for Real Symmetric Matrices)

Let $A \in \mathbb{R}^{n \times n}$ be symmetric ($A = A^T$). Then A has n real eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda^n$ (counted with multiplicity) and n orthonormal eigenvectors $q_1, \ldots, q^n$ such that:

$$A = Q \Lambda Q^T = \Sigma_i \lambda_i q_i q_i^T$$

where $Q = [q_1 \mid \ldots \mid q^n] \in O(n)$ is orthogonal and $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda^n)$. For symmetric matrices, the SVD coincides with the eigendecomposition: singular values equal absolute eigenvalues $|\lambda_i|$, and left and right singular vectors coincide (up to sign flips for negative eigenvalues).

Positive definiteness. A symmetric matrix A is positive definite ($A \succ 0$) if $x^T Ax > 0$ for all $x \neq 0$, equivalently if all eigenvalues are strictly positive. It is positive semidefinite ($A \succcurlyeq 0$) if $x^T Ax \geq 0$, equivalently if all eigenvalues are nonnegative. Positive definite matrices define inner products: $\langle x, y \rangle_a = x^T Ay$. In optimisation, a twice-differentiable function f is locally convex at θ if and only if its Hessian $H = \nabla^2 f(\theta)$ is positive semidefinite. The condition number $\kappa(A) = \lambda_1 / \lambda^n$ of a SPD matrix measures how elongated the associated ellipsoid is; high condition number indicates an ill-conditioned quadratic form where gradient descent converges slowly because directions of high curvature and low curvature require different step sizes.

The matrix square root of a SPD matrix A is the unique SPD matrix A^{1/2} = Q Λ^{1/2} Q^T satisfying A^{1/2} A^{1/2} = A. The inverse A^{-1} = Q Λ^{-1} Q^T exists when A $\succ 0$. The Cholesky decomposition A = LL^T (L lower triangular with positive diagonal) is the computationally efficient factorisation for solving linear systems involving SPD matrices. All these factorisations will appear repeatedly in the treatment of natural gradient methods in Chapter 4.

1.4 Random Matrix Theory and the Marchenko-Pastur Law

Neural network weight matrices are initialised from probability distributions (Glorot, He, orthogonal) and evolve under stochastic gradient dynamics. Random Matrix Theory (RMT) provides the appropriate null model: the distribution of singular values one would expect if the weight matrix had no learned structure. Deviations from this null model identify structure that has been extracted from data.

The fundamental RMT result for rectangular matrices is the Marchenko-Pastur law, which describes the empirical spectral distribution (ESD) of the sample covariance matrix of a random matrix in the limit of large dimensions.

Theorem 1.4 (Marchenko-Pastur Law)

Let $X \in \mathbb{R}^{n \times p}$ have i.i.d. entries with mean 0 and variance σ^2, and let $\gamma = p/n \to c \in (0, \infty)$ as $n, p \to \infty$. The empirical spectral distribution of $(1/p)XX^T$ converges almost surely (in the weak sense) to the Marchenko-Pastur distribution with density:

$$\rho(\lambda) = (1/(2\pi\gamma\sigma^2)) \times \sqrt{[(b - \lambda)(\lambda - a)]} / \lambda \quad \text{for } \lambda \in [a, b]$$

where the support boundaries are $a = \sigma^2(1 - \sqrt{\gamma})^2$ and $b = \sigma^2(1 + \sqrt{\gamma})^2$. When $\gamma < 1$ (more samples than features), there is a point mass of weight $1 - 1/\gamma$ at 0. The distribution has mean σ^2 and reflects the universal fact that noise in high dimensions has a predictable spectral shape determined only by the aspect ratio γ.

Implications for deep learning. Any singular value of a weight matrix that falls within the Marchenko-Pastur bulk [a, b] is statistically indistinguishable from noise and represents a direction in which the weight matrix behaves like

a random projection. Singular values outside this bulk — either spiked eigenvalues (Martin-Mahoney heavy-tail hypothesis) or values below the lower edge — represent structure that has been extracted from training data. This provides a principled criterion for model compression: prune singular values within the bulk without loss of meaningful learned structure.

The Heavy-Tail Self-Regularisation (HT-SR) theory, developed by Martin and Mahoney, extends this picture beyond the Marchenko-Pastur null model. Empirical studies of trained networks across architectures reveal that the ESD of well-generalising weight matrices follows a power law $\varrho(\lambda) \propto \lambda^{-\alpha}$ with exponent $\alpha \in (2, 4)$. Networks in this range exhibit what the authors call implicit self-regularisation: the training dynamics naturally produce a spectrum that deviates from the Gaussian baseline in a way that correlates with held-out accuracy. This phenomenon — that training dynamics have an implicit spectral regularisation effect — is one of the most striking empirical regularities in the study of large neural networks and provides a connection between RMT, statistical physics, and practical model quality assessment.

Regime	ESD Shape	Interpretation	Network Quality
Gaussian/untrained	Marchenko-Pastur bulk	Pure noise, no learned structure	Random (chance performance)
$\alpha > 4$ (Weak power law)	Slight heavy tail	Mild learning, marginal structure	Below average generalisation
$\alpha \in (2, 4)$ (Power law)	Heavy-tailed, spike above bulk	Strong feature extraction	Good generalisation
$\alpha < 2$ (Bulk+spike)	Rank-1 perturbation	Potential overfit / specialised features	Task-dependent
Flat/rank-deficient	Concentrated mass at 0	Effective low-rank collapse	May indicate training failure

1.5 Low-Rank Structure and LoRA: The SVD Perspective

The most practically consequential application of SVD to modern deep learning is Low-Rank Adaptation (LoRA), a parameter-efficient fine-tuning method that has become the dominant approach for adapting large pre-trained language models to downstream tasks. LoRA's effectiveness rests directly on an empirical observation with an SVD explanation: the intrinsic dimensionality of fine-tuning updates is far lower than the ambient parameter dimension.

Setup. Let $W_0 \in \mathbb{R}^{m \times n}$ be a frozen pre-trained weight matrix. Standard fine-tuning learns $\Delta W = W - W_0$ by gradient descent on a task-specific loss L. LoRA instead parameterises the update as $\Delta W = BA$ where $B \in \mathbb{R}^{m \times r}$, $A \in \mathbb{R}^{r \times n}$, and $r \ll \min(m, n)$ is a hyperparameter called the intrinsic rank. During training, W_0 is frozen and only B and A are updated. At inference, the adapted weight is $W = W_0 + (s/r)\, BA$, where s is a scaling factor (often set equal to r, making $s/r = 1$, or to the original learning rate α, in which case $s/r = \alpha/r$).

Parameter count. Standard fine-tuning requires storing and updating mn gradients per weight matrix. LoRA requires only r(m+n) parameters. For typical values $m = n = 4096$ and $r = 8$, this is a compression factor of $mn / r(m+n) \approx n/(2r) = 256$. Across a full transformer with many weight matrices, total trainable parameter count drops from billions to tens of millions.

Why does this work? The SVD offers a compelling explanation. Suppose the optimal update ΔW^* has effective rank k (in the sense that its top-k singular values account for nearly all its Frobenius norm). Then by Eckart-Young-Mirsky, ΔW^* is well-approximated by its rank-r truncation for any $r \geq k$. LoRA implicitly searches the space of rank-r matrices for the best approximation to ΔW^*. If k is small — as empirically appears to be the case for many language tasks — then $r \geq k$ suffices to capture essentially all of the task-specific update. Hu et al. (2021) verified this by showing that the intrinsic dimension of fine-tuning on GLUE benchmarks is often below 100, orders of magnitude smaller than the weight matrix dimensions.

Theorem 1.5 (LoRA Approximation Quality)

Let ΔW^* be the optimal weight update and let its SVD be $\Delta W^* = U\Sigma V^T$. Let $\Sigma = diag(\sigma_1, ..., \sigma_r)$ with $\sigma_1 \geq ... \geq \sigma_r \geq 0$ and let $\Sigma_r = diag(0,...,0, \sigma_{r+1},...,\sigma_r)$ contain only singular values beyond rank r. Then the LoRA approximation $\Delta W = BA$ achieves:

$min^{N,M} \|\Delta W^* - BA\|_t = \|\Sigma_r\|_t = \sqrt{(\sigma_{r+1}^2 + ... + \sigma_r^2)}$

This approximation error decreases as r increases and is zero if $r \geq rank(\Delta W^*)$. The practical implication is that r should be chosen at the knee of the singular value curve of ΔW^*: the smallest r such that the excluded singular values are small. Empirically, $r \in \{4, 8, 16, 32\}$ is sufficient for most NLP fine-tuning tasks.

1.6 Tensor Decompositions and Higher-Order Structure

Weight matrices are second-order tensors. Many deep learning models involve higher-order tensors: convolutional filters W ∈ ℝ^{C_out × C_in × k × k}, attention weight tensors, and parameter tensors in multi-head attention. The generalisation of SVD to higher-order tensors introduces several non-equivalent notions of rank and factorisation.

Tucker decomposition. A tensor $\Omega \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ admits a Tucker decomposition $\Omega = G \times_1 U^1 \times_2 U^2 \times_3 U^3$, where $G \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ is the core tensor and $U^{(k)} \in \mathbb{R}^{d_k \times r_k}$ are the factor matrices. The multilinear ranks (r_1, r_2, r_3) play the role of SVD rank. Tucker decomposition is computed via Higher-Order SVD (HOSVD): unfold the tensor along each mode, compute the SVD of the resulting matrix, and take the leading left singular vectors as factor matrices. Tucker decomposition for convolutional filters enables compression: a filter with $r_1 = r_2 = r_3 = r$ reduces the parameter count from C_out C_in k^2 to $r_1 r_2 r_3$ + C_out r_1 + C_in r_2 + 2k r_3.

CP decomposition. The canonical polyadic (CP) decomposition writes $\Omega = \sum_{i=1}^{r} \lambda_i a_i \otimes b_i \otimes c_i$, a sum of rank-one tensors. The CP rank of Ω is the minimum r for which such a decomposition exists. Unlike the matrix case, computing CP rank is NP-hard in general, and best low-CP-rank approximation does not always exist (the problem is not always attained). Despite these theoretical challenges, CP decomposition is widely used for recurrent weight tensor

compression and for parameterising multi-head attention in memory-efficient settings.

The key distinction for practical purposes is that Tucker decomposition is numerically stable and admits a convergent algorithm (HOSVD followed by higher-order orthogonal iteration, HOOI), whereas CP decomposition has no analogously stable algorithm in full generality. The choice between them represents a tradeoff: Tucker better preserves the multilinear structure of the tensor but requires more parameters than CP for the same effective rank.

Chapter 2: Backpropagation as Automatic Differentiation

Backpropagation is the algorithm that makes large-scale neural network training computationally feasible. It is not, as is sometimes implied by introductory treatments, a mysterious domain-specific trick. It is the chain rule of differential calculus applied systematically to a directed acyclic computation graph, executed in reverse topological order. This reframing — as a special case of the general theory of automatic differentiation (AD) — is not merely an intellectual tidying. It opens connections to the theory of differentiable programming, reveals the precise computational cost of gradient computation, and clarifies the relationship between backpropagation and the higher-order differentiation required for curvature-aware optimisers.

2.1 Computation Graphs and the Chain Rule

A computation graph G = (V, E) is a directed acyclic graph where nodes represent values (tensors) and edges represent functional dependencies. Each non-input node v has an associated elementary function f_v that maps its parents' values to its own value: val(v) = f_v(val(pa(v))). The composition of these elementary functions defines the network's overall function.

Formal setup. Let the forward computation produce intermediate values a_1, a_2, ..., a^k from inputs x and parameters θ. Denote the scalar loss as $L = a^k$. We seek the gradient $\nabla\theta\ L = \partial L/\partial\theta$ for parameter update, and $\nabla^x\ L = \partial L/\partial x$ for, e.g., input gradient methods or architectural differentiation. The chain rule gives the fundamental recursion: for any intermediate variable a_k appearing in the path from θ to L:

$$\partial L/\partial a_k = \Sigma_j{:}\ a_k \in pa(a_j)\ (\partial L/\partial a_j) \times (\partial a_j/\partial a_k)$$

This is the general form of the backward pass. The quantities $\delta_k \equiv \partial L/\partial a_k$ are the error signals (or sensitivity, or upstream gradient) at each node.

Backpropagation computes all of them in a single backward pass by traversing the computation graph in reverse topological order.

2.2 Forward Mode and Reverse Mode Automatic Differentiation

The theory of automatic differentiation distinguishes two canonical modes of propagating derivatives through a computation graph: forward mode (tangent mode) and reverse mode (adjoint mode, cotangent mode). Neural network backpropagation is reverse-mode AD. Understanding both modes and their computational complexity is essential for appreciating why backpropagation is computationally efficient and when alternatives are warranted.

Forward mode AD. Forward mode propagates directional derivatives alongside the forward computation. For a chosen tangent direction $d\theta \in \mathbb{R}^p$ (a unit vector in parameter space), forward mode computes, for each node v: $\dot{v} \equiv dval(v)/dt \mid_{t=0}$ *where* $\theta(t) = \theta + t\, d\theta$. The update rule is the Jacobian-vector product (JVP): $\dot{v} = Jf_v \cdot \{pa(v)\}$, applied in forward topological order. A single forward pass yields the directional derivative $dL/dt = \nabla\theta\, L \cdot d\theta$. To recover the full gradient $\nabla\theta\, L \in \mathbb{R}^p$, forward mode requires P separate forward passes (one per parameter direction), costing $O(P)$ forward-pass compute. For $P \gg 1$ outputs, forward mode is efficient; for $P \gg 1$ parameters and a scalar loss, it is not.

Reverse mode AD. Reverse mode computes all partial derivatives $\partial L/\partial a_k$ in a single backward pass by propagating the scalar cotangent $\delta = 1$ from the output node backward. The update rule is the vector-Jacobian product (VJP): $\delta^p{}_a(v) \cong \delta v \times Jf_v$, accumulated into each parent's cotangent. Because the loss L is a scalar ($P = 1$ output), a single backward pass recovers the full gradient $\nabla\theta\, L$ at cost $O(1)$ times the forward pass. This is why reverse-mode AD (backpropagation) is the standard for neural network training: the ratio (outputs)/(parameters) is $1/P \ll 1$, and reverse mode's cost scales with outputs rather than parameters.

Theorem 2.1 (Complexity of Reverse-Mode AD)

Let $f: \mathbb{R}^p \to \mathbb{R}$ be computed by a program with T arithmetic operations. Reverse-mode AD computes ∇f in at most cT arithmetic operations for a small universal constant $c \leq 6$. This bound is tight: there exist functions where $c = 1$ (addition), and the constant c arises from the overhead of the backward pass's VJP primitives. The key implication is that computing the full gradient of a scalar loss is never more than a constant factor more expensive than a forward pass, regardless of the number of parameters P.

Corollary (Memory Cost): Reverse-mode AD requires storing all intermediate activations for the backward pass. A network with L layers and intermediate activation size $O(N)$ per layer requires $O(LN)$ memory for the full backward pass. Gradient checkpointing trades time for memory by recomputing activations during the backward pass, reducing memory to $O(\sqrt{L} \times N)$ at the cost of one additional partial forward pass.

2.3 Jacobians, Gradient Computation, and the Layer-by-Layer Analysis

For a neural network with L layers, each performing the computation $a_l = \sigma(W_l a_{l-1} + b_l)$, the gradient of the loss L with respect to the weight matrix W_l follows a precise pattern that reveals the vanishing gradient problem.

Gradient of a single layer. Define the pre-activation $z_l = W_l a_{l-1} + b_l$ and the post-activation $a_l = \sigma(z_l)$ (applied elementwise). The error signal at layer ℓ is $\delta_l = \partial L / \partial z_l \in \mathbb{R}^{n_\ell}$. Backpropagation computes:

$$\delta_l = (W_{l+1}^T \delta_{l+1}) \odot \sigma'(z_l)$$

$$\nabla_{W_l} L = \delta_l (a_{l-1})^T \quad \textit{(outer product)}$$

The gradient $\nabla_{W_l} L$ is an outer product of the local error signal and the incoming activation. Its Frobenius norm satisfies $\|\nabla_{W_l} L\|_t = \|\delta_l\| \times \|a_{l-1}\|$, factoring the upstream error magnitude (propagated through W_{l+1}) and the local input magnitude.

The vanishing gradient problem. The error signal at layer ℓ is related to the output error by the product of Jacobians of intermediate layers: $\delta_l = J_{L \to l} \delta_L$

where $J_{L\to l} = \prod_{k=l+1}^{L} D_k W_k$, and $D_k = \mathrm{diag}(\sigma'(z_k))$ is the diagonal matrix of activation derivatives. If all W_k have spectral norm $\|W_k\|_2 = \varrho$ and σ' is bounded by λ (e.g., $\lambda = 1/4$ for sigmoid), then $\|\delta_l\| \le (\varrho\lambda)^{L-l} \|\delta_L\|$. For sigmoid activations with $\varrho \approx 1$, this gives exponential decay at rate $(\lambda)^{L-l} = (1/4)^{L-l}$, causing gradients to vanish exponentially in the depth $L - \ell$.

Theorem 2.2 (Exponential Gradient Decay)

Consider an L-layer network with weight matrices W_l and activation function σ with $\|\sigma'\|_\infty \le \lambda$. If $\|W_l\|_2 \le \rho$ for all ℓ, then the gradient at layer ℓ satisfies:

$$\|\nabla_{W_l} L\|_t \le (\rho\lambda)^{L-l} \times C$$

where C depends on the output gradient magnitude and the activations at layer ℓ. For sigmoid activations ($\lambda = 1/4$) and weight matrices with spectral norm near 1, this bound decays exponentially in $L - \ell$, causing gradients to become negligible for early layers of deep networks. The exploding gradient regime occurs symmetrically: if $\rho\lambda > 1$, gradients grow exponentially, destabilising training. The critical stability condition $\rho\lambda \approx 1$ is the principle underlying careful weight initialisation schemes.

2.4 Residual Connections, Highway Networks, and Gradient Highways

The architectural solution to the vanishing gradient problem — residual connections — is one of the most consequential ideas in the history of deep learning. Introduced by He et al. in 2015, the residual connection transforms the layer update from $a_l = F(a_{l-1}; W_l)$ (unrestricted) to:

$$a_l = F(a_{l-1}; W_l) + a_{l-1}$$

The identity shortcut allows gradients to bypass the nonlinear sublayer entirely. The Jacobian of the residual block with respect to a_{l-1} is:

$$\partial a_l / \partial a_{l-1} = I + \partial F / \partial a_{l-1}$$

The identity term I means that the spectral norm of the block Jacobian is bounded below by 1 (since $\|I + J\|_2 \ge 1$ for any J). This prevents gradient vanishing: even if the residual function F learns nearly zero ($J \approx 0$), the gradient

still propagates through the identity shortcut with coefficient 1. More precisely, the gradient of the loss with respect to the early layer activation a_l satisfies:

Theorem 2.3 (Residual Network Gradient Flow)

For a residual network $a_l = a_{l-1} + F_l(a_{l-1})$, the gradient of the loss L with respect to the activation a_l satisfies (by unrolling the chain rule through blocks $i = \ell, \ldots, L$):

$$\partial L/\partial a_l = (1 + \Sigma_{k=l+1}^{L} \prod_{i=l+1}^{k} (I + \partial F_i/\partial a_{i-1})) \times \partial L/\partial a_L$$

The sum contains a constant term 1 from the unimpeded path through all identity shortcuts. This means that the gradient at layer ℓ always contains a term equal to the output gradient $\partial L/\partial a_L$ multiplied by the identity, providing a non-vanishing gradient signal regardless of depth L and regardless of what the residual functions F_i have learned. This is the fundamental mechanism behind the trainability of very deep (100+ layer) residual networks.

The highway network (Srivastava et al., 2015), proposed contemporaneously, uses a learned gating mechanism: $a_l = T(a_{l-1}) \odot F(a_{l-1}) + (1 - T(a_{l-1})) \odot a_{l-1}$, where $T \in [0,1]^n$ is a sigmoid-gated carry. The highway network strictly generalises the residual connection ($T = 1$ recovers the non-residual case; $T = 0$ the identity shortcut). The residual connection — which imposes $T = 0.5$ implicitly and without extra parameters — proved more effective in practice, likely because the fixed identity shortcut provides a guaranteed gradient path without the additional expressivity that the gate introduces.

2.5 Normalisation Layers: Batch, Layer, Group, and RMS Normalisation

Normalisation layers are architectural interventions that stabilise the distribution of activations during forward propagation, with direct consequences for the backward pass. Unlike residual connections, which address gradient magnitude, normalisation addresses gradient direction: it prevents the internal covariate shift that occurs when the distribution of activations shifts as parameters update, causing later layers to perpetually adapt to a moving target.

Batch Normalisation (BatchNorm). For a batch of activations $\{x_1, ..., x_B\}$ at a given layer, BatchNorm computes the normalised activation:

$$\mu_B = (1/B) \Sigma_i x_i, \quad \sigma_B^2 = (1/B) \Sigma_i (x_i - \mu_B)^2 + \varepsilon$$

$$\hbar_i = (x_i - \mu_B) / \sigma_B, \quad y_i = \gamma \hbar_i + \beta$$

The learnable parameters γ (scale) and β (shift) allow the layer to undo the normalisation if that is optimal. BatchNorm's gradient with respect to x_i has the critical property of being orthogonal to both the mean direction and the scale direction: the normalised activations $\hbar_i$ have zero mean and unit variance by construction, and gradients through the normalisation are projected onto the null space of the centering and scaling operations. This orthogonality removes two degrees of freedom per channel from the gradient, effectively preconditioning the optimisation by eliminating directions that merely correspond to changes in the mean or variance of activations.

Layer Normalisation. LayerNorm normalises across the feature dimension for each sample independently: $\mu_i = (1/d) \Sigma_j x_{ik}$, $\sigma_i^2 = (1/d) \Sigma_j (x_{ik} - \mu_i)^2$. Unlike BatchNorm, it does not depend on the batch, making it suitable for variable-length sequences and small batches. The Pre-LN variant (LayerNorm applied before each sublayer, as in GPT-2 and most modern transformers) has a particularly clean gradient analysis:

Theorem 2.4 (Pre-LN Gradient Norm Bound)

For a Pre-LN Transformer with L blocks, where each block is $x_{\ell+1} = x_\ell + F_\ell(LN(x_\ell))$, the gradient of the loss L with respect to the weight matrix W_l in block ℓ satisfies:

$\|\nabla_{W_l} L\|_t \leq C / \sqrt{L}$

where C is a constant that depends on the loss gradient magnitude and the Lipschitz constant of F but is independent of depth L. This bound guarantees that gradients in early layers do not vanish (lower-bounded by a quantity that decays only as $1/\sqrt{L}$ rather than exponentially) and do not explode (upper-bounded by the same rate). The $\sqrt{L}$ scaling explains why Pre-LN models train stably without learning rate warm-up: the gradient scale across layers is approximately uniform even at the start of training.

RMSNorm. Root Mean Square Normalisation (Zhang and Sennrich, 2019) simplifies LayerNorm by removing the mean-centering step: $y_i = x_i / \text{RMS}(x_i) \times \gamma$, where $\text{RMS}(x) = \sqrt{((1/d)\Sigma_k\ x_k^2)}$. The motivation is that re-centering (subtracting the mean) is computationally wasteful and empirically less important than re-scaling. RMSNorm is now the default in Llama and related models, saving a small but non-negligible fraction of compute per layer in large models.

2.6 Second-Order Information: Hessians, Gauss-Newton, and Curvature

The gradient $\nabla\theta\ L$ provides first-order information about the loss landscape. The Hessian $H = \nabla^2\theta\ L$ provides second-order (curvature) information. Exploiting curvature can dramatically improve optimisation, but the Hessian of a modern neural network has P^2 entries (where P is in the billions) and is completely intractable to store or invert. This motivates the family of approximate second-order methods discussed in Chapter 4.

Computing Hessian-vector products. Although the full Hessian $H \in \mathbb{R}\text{^}\{P\times P\}$ is intractable, the Hessian-vector product Hv for a given vector $v \in \mathbb{R}^P$ can be computed in O(P) time by applying reverse-mode AD twice. Specifically, $Hv = \nabla\theta(\nabla\theta\ L \cdot v)$, which requires one forward and two backward passes. This enables iterative methods (conjugate gradient, Lanczos) that access H only through matrix-vector products, and forms the computational basis for second-order methods like Newton-CG and natural gradient with iterative linear solvers.

The Gauss-Newton matrix. For a loss of the form $L(\theta) = \Sigma_i\ \ell(f_\theta(x_i), y_i)$, the Gauss-Newton matrix $G = J^T\ H_\ell\ J$ (where $J = \partial f_\theta/\partial\theta$ is the Jacobian and $H_\ell = \partial^2\ell/\partial f^2$ is the Hessian of the loss w.r.t. outputs) is a positive semidefinite approximation to the Hessian that ignores the second-derivative terms from the network's nonlinearity. For squared error loss, G coincides exactly with the Fisher information matrix (see Chapter 3). For cross-entropy loss with softmax output, $G \approx F$ (the Fisher) up to approximation error that vanishes near

convergence. The Gauss-Newton matrix is positive semidefinite by construction ($G = J^T H_\ell J$ with $H_\ell \succcurlyeq 0$), making Newton steps $G^{-1}\nabla L$ well-defined as a descent direction even when the Hessian has negative curvature directions.

Chapter 3: Information Geometry and the Fisher Metric

The classical development of optimisation for neural networks treats the parameter space $\theta \in \mathbb{R}^p$ as a flat Euclidean space and uses the Euclidean gradient $\nabla_\theta L$ as the direction of steepest descent. This choice is not wrong — it leads to algorithms that converge — but it ignores a fundamental structure of the problem: the parameters θ are not abstract real numbers but the parameters of a probability distribution $p(y \mid x; \theta)$, and the natural geometry on the space of probability distributions is not Euclidean. Information geometry, the mathematical framework developed by Shun-ichi Amari from the 1980s onward, provides the correct geometric structure and leads to optimisation algorithms that are provably faster and more robust than Euclidean gradient descent.

3.1 Statistical Manifolds: The Space of Probability Distributions

Consider a parametric family of probability distributions $\in = \{p(x; \theta) : \theta \subset \Theta \circledcirc \mathbb{R}^p\}$, where each distribution is a function $x \mapsto p(x; \theta)$ from the sample space Ω to $[0, 1]$ satisfying $\int p(x; \theta)\, dx = 1$. Under regularity conditions (smoothness in θ, support independent of θ, identifiability), $\mathscr{M}$ is a smooth manifold of dimension P, and each distribution $p(\cdot; \theta)$ is a point on this manifold.

The question that information geometry addresses is: what is the natural notion of distance between two nearby distributions $p(\cdot; \theta)$ and $p(\cdot; \theta + d\theta)$? The answer is not the Euclidean distance $\|d\theta\|$, which depends on the arbitrary choice of parameterisation, but a geometry-invariant distance derived from the statistical structure of the family.

KL divergence and its second-order approximation. The Kullback-Leibler divergence $D_{KL}(p \| q) = \int p(x) \log(p(x)/q(x))\, dx$ measures the asymmetric "distance" from q to p. It is not a metric (not symmetric, does not satisfy the

triangle inequality), but its second-order Taylor expansion around p ≈ q gives a metric structure. For distributions p(·; θ) and p(·; θ + dθ):

$$D_{KL}(p(\theta) \| p(\theta + d\theta)) = (1/2)\, d\theta^T\, G(\theta)\, d\theta + O(\|d\theta\|^3)$$

where G(θ) is the Fisher Information Matrix (FIM). The symmetric KL divergence D_{KL}(p‖q) + D_{KL}(q‖p) has the same second-order expansion with coefficient 1 (no factor of 1/2). This shows that the FIM is precisely the matrix that makes the KL divergence — the natural measure of distributional difference in statistics — correspond to a Riemannian metric on the parameter manifold.

3.2 The Fisher Information Matrix as a Riemannian Metric

The Fisher Information Matrix (FIM) is the central object of information geometry. Its multiple equivalent definitions reveal its deep connections to statistics, information theory, and optimisation.

Definition 3.1 (Fisher Information Matrix)

For a parametric family {p(x; θ)}, the Fisher Information Matrix G(θ) ∈ ℝ^{P×P} is defined equivalently as:

(Score form) G_{ij}(θ) = E_{p(x;θ)}[∂_i log p(x;θ) × ∂_j log p(x;θ)]

(Hessian form) G_{ij}(θ) = -E_{p(x;θ)}[∂_i∂_j log p(x;θ)]

(Outer product form) G(θ) = E_{p(x;θ)}[s(x;θ) s(x;θ)^T] where s(x;θ) = ∇θ log p(x;θ) is the score function

The equivalence of the score and Hessian forms follows from Fisher's identity: E[s] = 0 (the score has zero mean under its own distribution), which implies E[∂_i s_j] = -E[s_i s_j]. G(θ) is always positive semidefinite (as an outer product expectation) and positive definite for identifiable, regular models. It transforms covariantly under reparameterisation: if η = Φ(θ), then G_η(η) = JΦ^{-T} G_θ(θ) JΦ^{-1}, making the inner product dθ^T G(θ) dθ = dη^T G(η) dη invariant.

The Cramér-Rao lower bound. The FIM is intimately connected to the fundamental limits of statistical estimation. The Cramér-Rao inequality states

that for any unbiased estimator $\hat{\theta}(x)$ of θ based on a single observation $x \sim p(\cdot; \theta^*)$, the covariance matrix satisfies:

Theorem 3.1 (Cramér-Rao Lower Bound)

Let $\hat{\theta}(x)$ be any unbiased estimator of θ. Then $Cov(\hat{\theta}) = E[(\hat{\theta} - \theta)(\hat{\theta} - \theta)^T] \succeq G(\theta)^{-1}$.

Equivalently, for any direction $v \in \mathbb{R}^p$: $Var(v^T \hat{\theta}) \geq v^T G(\theta)^{-1} v$.

The bound is attained by the maximum likelihood estimator (MLE) in the limit of large sample size n: $n^{1/2}(\hat{\theta}_{MLE} - \theta^*) \to \mathcal{N}(0, G(\theta^*)^{-1})$ in distribution. This makes the FIM the variance of the best possible estimator, and its inverse G^{-1} the achievable precision for any estimator. The natural gradient descent direction $G^{-1}\nabla L$ can thus be interpreted as moving in the direction that most efficiently improves the statistical fit, measured in units of the minimum achievable estimation variance.

3.3 Exponential Families and Their Natural Geometry

The class of exponential family distributions has a remarkably clean information-geometric structure. An exponential family is defined by:

$$p(x; \eta) = h(x) \exp(\langle \eta, T(x) \rangle - A(\eta))$$

where $\eta \in \mathbb{R}^p$ are the natural parameters, $T(x) \in \mathbb{R}^p$ is the sufficient statistic, $h(x)$ is the base measure, and $A(\eta) = \log \int h(x) \exp(\langle \eta, T(x) \rangle)\, dx$ is the log-partition function. The exponential family includes Gaussians, Bernoulli, Poisson, Gamma, Beta, and Dirichlet distributions as special cases, and it is the natural output family for neural networks (the final layer's softmax corresponds to the Bernoulli/Categorical exponential family with natural parameters equal to the logits).

Dual parameterisations and Bregman divergences. The log-partition function $A(\eta)$ is strictly convex, and its gradient defines the mean parameter $\mu = \nabla A(\eta) = E_\eta[T(x)]$. The Legendre transform $A^*(\mu) = \langle \eta, \mu \rangle - A(\eta)$ defines the dual potential (negative entropy). The FIM of an exponential family is $\nabla^2 A(\eta)$, the Hessian of the log-partition function, and is positive definite by the strict convexity of A. This means that the information geometry of exponential

families is induced by a Bregman divergence: $D_{KL}(p(\eta') \| p(\eta)) = B_A(\eta' \| \eta) = A(\eta') - A(\eta) - \langle \nabla A(\eta), \eta' - \eta \rangle$, the Bregman divergence generated by A.

This structure makes exponential families the preferred output families for neural networks not only because they include the most common distributions, but because their information geometry is computationally tractable: the FIM equals the Hessian of a convex function ($\nabla^2 A$), and natural gradient descent reduces to steepest descent with respect to the Bregman divergence. For a neural network with softmax output and cross-entropy loss, the Fisher of the output layer is:

$$G_{out} = E_{y \sim p(\cdot;\theta)}[\nabla\theta \log p(y;\theta) \, \nabla\theta \log p(y;\theta)^T]$$

which for the softmax equals $\mathrm{diag}(p) - pp^T$ (the covariance matrix of the prediction distribution), a matrix that concentrates near zero when the network is confident and spreads when uncertain. This connection between confidence and curvature is operationally important: natural gradient descent takes smaller steps in directions where the network is already confident and larger steps in directions where there is uncertainty about the output.

3.4 The Alpha-Connections and Dually Flat Manifolds

A Riemannian manifold (M, g) can be equipped with multiple affine connections ∇, each defining a different notion of parallel transport and geodesic. Amari's central insight is that the statistical manifold S carries a one-parameter family of affine connections $\nabla^{(\alpha)}$, parameterised by $\alpha \in \mathbb{R}$, each of which is the natural connection for a different inference problem.

The α-connections. The α-connection $\nabla^{(\alpha)}$ is defined by its Christoffel symbols:

$$\Gamma^{(\alpha)}_{ij,k} = E_p[(\partial_i \partial_j \ell_\theta + ((1-\alpha)/2) \, \partial_i \ell_\theta \, \partial_j \ell_\theta) \, \partial_k \ell_\theta]$$

where $\ell_\theta = \log p(x;\theta)$. The two extremes are the most important: $\alpha = +1$ gives the e-connection (exponential connection), whose geodesics are exponential family subfamilies; $\alpha = -1$ gives the m-connection (mixture connection), whose geodesics are mixture model subfamilies. The $\alpha = 0$ connection is the Levi-

Civita connection of the Fisher metric (the unique torsion-free metric-compatible connection). The critical structural fact is:

Theorem 3.2 (Duality of e- and m-Connections)

The e-connection ($\alpha = +1$) and m-connection ($\alpha = -1$) are dual with respect to the Fisher metric: for any smooth curve $\gamma(t)$ and vector fields X, Y along γ,

$$\partial_t \langle X, Y \rangle_G = \langle \nabla^{(+1)}_{\dot\gamma} X, Y \rangle_G + \langle X, \nabla^{(-1)}_{\dot\gamma} Y \rangle_G$$

For exponential families, the e-connection is flat (zero curvature) in natural parameters η and the m-connection is flat in mean parameters μ. Amari calls this structure a dually flat manifold. The consequence is that both the e-geodesic projection (maximum likelihood over an exponential family submodel) and the m-geodesic projection (moment matching, as in the E-step of EM) have unique global solutions characterised by Pythagorean theorems: for a projection p^* of q onto a submanifold M,

$$D_{KL}(p \,\|\, q) = D_{KL}(p \,\|\, p^*) + D_{KL}(p^* \,\|\, q) \quad \text{[m-geodesic projection]}$$

$$D_{KL}(q \,\|\, p) = D_{KL}(q \,\|\, p^*) + D_{KL}(p^* \,\|\, p) \quad \text{[e-geodesic projection]}$$

These Pythagorean identities are the foundation of the convergence theory for EM, variational inference (ELBO maximisation as m-projection), and natural gradient descent (e-geodesic step).

3.5 The Information Bottleneck and Mutual Information in Deep Networks

The Information Bottleneck (IB) principle, introduced by Tishby, Pereira, and Bialek (1999), provides an information-theoretic objective for learning representations. Given input X and label Y, the IB seeks a stochastic encoding Z = f_θ(X) that preserves information about Y while compressing information about X:

$$\min_{p(z|x)} \left[-I(Z; Y) + \beta\, I(X; Z) \right]$$

The Lagrange multiplier $\beta \in [0, \infty)$ controls the compression-accuracy tradeoff. The IB optimum at each β lies on the Information Plane — the plot of I(Z; Y) versus I(X; Z) — on the convex hull of achievable points, the IB curve. The self-consistent equations for the optimal encoder are:

$$p^*(z|x) \propto p(z) \exp(\beta\, D_{KL}(p(y|x) \,\|\, p(y|z)))$$

This is a Boltzmann distribution where the energy is the KL divergence between the conditional output distribution and the expected conditional, weighted by β. The IB equations can be solved iteratively (IB algorithm, an EM-like procedure) or approximated using deep networks as the encoder-decoder pair (Deep VIB, Alemi et al. 2017).

The IB and mutual information estimation. A controversial but influential claim by Tishby and Schwartz-Ziv (2017) suggested that neural network layers spontaneously traverse the information plane during training, exhibiting a fitting phase ($I(Z; Y)$ increases) followed by a compression phase ($I(X; Z)$ decreases). Subsequent work (Saxe et al., 2018; Goldfeld et al., 2019) showed that this phenomenology is highly sensitive to the choice of activation function and mutual information estimator. Networks with ReLU activations show little to no compression phase in deterministic settings; binning estimators of mutual information introduce artefacts. Nevertheless, the IB framework remains a valuable conceptual tool for thinking about what an optimal representation should look like and for designing training objectives that explicitly balance informativeness and compression.

Chapter 4: Natural Gradient Descent and K-FAC

The Fisher Information Matrix defines the Riemannian metric on the statistical manifold of the model's output distributions. Steepest descent on this manifold — the natural gradient — is the optimal first-order update rule in the sense of being parameterisation-invariant and locally optimal in KL divergence. But the FIM of a deep neural network is a P×P matrix for P in the billions: intractable to store, let alone invert. The history of practical natural gradient methods is the history of increasingly accurate tractable approximations to this intractable matrix. Kronecker-Factored Approximate Curvature (K-FAC) is the most successful such approximation, achieving second-order-like convergence rates at a cost competitive with first-order methods.

4.1 Steepest Descent on Riemannian Manifolds

In Euclidean space with the standard inner product, the gradient $\nabla\theta\ L$ is the direction of steepest ascent of L at θ: the unit vector that maximises the directional derivative $dL/d\theta \cdot v$ subject to $\|v\| = 1$. Gradient descent moves in the opposite direction: $\theta \leftarrow \theta - \eta\ \nabla\theta\ L$.

On a Riemannian manifold (M, G), the steepest ascent direction is the unit vector that maximises the directional derivative subject to the Riemannian norm constraint $\|v\|_{G^2} = v^T\ G(\theta)\ v = 1$. By the Cauchy-Schwarz inequality for the inner product $\langle u, v\rangle_G = u^T\ G\ v$:

$$max_\{v:\ v^T\ Gv \le 1\}\ (\nabla\theta\ L)^T\ v = \|\nabla\theta\ L\|_\{G\^\{-1\}\}$$

achieved at $v^* = G(\theta)\^\{-1\}\ \nabla\theta\ L\ /\ \|G(\theta)\^\{-1\}\ \nabla\theta\ L\|_G$. The natural gradient is defined as:

$$\tilde{g} \equiv G(\theta)\^\{-1\}\ \nabla\theta\ L$$

and the natural gradient descent update is $\theta \leftarrow \theta - \eta\ \tilde{g}$. The transformation from $\nabla\theta\ L$ to $\tilde{g}$ by multiplying with $G\^\{-1\}$ adjusts for the curvature of the statistical manifold: in directions where the Fisher metric assigns high

curvature (large G eigenvalue), the natural gradient is smaller than the Euclidean gradient; in directions of low curvature (small G eigenvalue), the natural gradient amplifies the Euclidean gradient. This is precisely the preconditioning effect that makes natural gradient descent more efficient than ordinary gradient descent in ill-conditioned settings.

4.2 Reparameterisation Invariance: The Defining Property

The most fundamental property of the natural gradient — the one that distinguishes it from all other gradient modifications including Adam, Adagrad, and RMSProp — is its invariance under smooth reparameterisation of the parameters.

Theorem 4.1 (Reparameterisation Invariance of Natural Gradient)

Let $\eta = \Phi(\theta)$ be a smooth diffeomorphism, so that $p(x; \theta) = p(x; \Phi^{-1}(\eta))$ as functions of data. Let $\tilde{\nabla}_\theta$ and $\tilde{\nabla}_\eta$ be the natural gradients computed in the θ and η parameterisations respectively. Under the reparameterisation, the gradient descent trajectory starting from the same distribution is identical: the natural gradient flow $\theta(t)$ (mapped to distribution space) is the same as the $\eta(t)$ flow. Explicitly:

$$J_\Phi^{-1}\, \tilde{\nabla}_\eta(\eta) = \tilde{\nabla}_\theta(\theta) \quad \text{where } J_\Phi = \partial\eta/\partial\theta$$

This holds because the Fisher metric transforms as $G_\eta = J_\Phi^{-T}\, G_\theta\, J_\Phi^{-1}$ and the loss gradient transforms as $\nabla_\eta L = J_\Phi^{-T}\, \nabla_\theta L$, so $G_\eta^{-1}\, \nabla_\eta L = J_\Phi\, G_\theta^{-1}\, \nabla_\theta L$.

In contrast, the Euclidean gradient fails this invariance: $\nabla_\eta L = J_\Phi^{-T}\, \nabla_\theta L$ depends on the Jacobian J_Φ, and gradient descent in η takes steps in a different direction (in distribution space) than gradient descent in θ. This is why the learning rate and gradient descent dynamics in one parameterisation do not transfer to another for Euclidean gradient methods, but do transfer for natural gradient methods.

The practical significance of reparameterisation invariance is that the natural gradient treats the learning problem as one of moving through distribution space, regardless of how the distribution is parameterised. For neural networks, this matters because the same function $p(y|x;\, \theta)$ can be parameterised in infinitely many ways (e.g., by doubling all weights and

halving biases, or using different normalisation conventions), and a well-behaved optimiser should not be sensitive to this choice. Empirically, networks trained with natural gradient methods are substantially less sensitive to the exact form of weight initialisation and to the scale of the learning rate, consistent with the theoretical prediction.

4.3 The Fisher Information Matrix for Neural Networks

For a neural network that outputs a distribution $p(y|x;\theta)$ (e.g., softmax categorical, or Gaussian with predicted mean), the FIM can be written as an expectation over the data distribution $q(x)$ and the model's own predicted distribution:

$$G(\theta) = E_{x\sim q,\, y\sim p(\cdot|x;\theta)}\left[\nabla_\theta \log p(y|x;\theta) \times \nabla_\theta \log p(y|x;\theta)^T\right]$$

Using the dataset average to estimate the expectation over $q(x)$ and sampling from the model to estimate the expectation over y, this is the empirical Fisher. The true Fisher uses the true label distribution $p^*(y|x)$, giving the cross-entropy Hessian interpretation. For a neural network with L layers and parameters $\theta = (W_1, b_1, \ldots, W_L, b_L)$, the FIM has block structure:

$$G = \left[G_{\alpha\beta}\right]_{\alpha,\beta \in \text{layers}} \quad \text{where } G_{\alpha\beta} = E\left[s_\alpha s_\beta^T\right]$$

and $s_\alpha = \partial \log p / \partial \theta_\alpha$ is the score with respect to layer-α parameters. Even ignoring the off-diagonal blocks, the diagonal block $G_{\alpha\alpha}$ for a single weight matrix $W_\ell \in \mathbb{R}^{m\times n}$ is itself an $mn \times mn$ matrix — storing it requires $O(m^2n^2)$ memory, and inverting it requires $O(m^3n^3)$ time. This is completely intractable for modern networks.

4.4 K-FAC: Kronecker-Factored Approximate Curvature

K-FAC (Martens and Grosse, 2015) approximates the FIM block G_ℓ for layer ℓ as a Kronecker product of two much smaller matrices. The key structural observation is that the score for layer ℓ has the form:

$$s_{W_\ell} = \nabla_{W_\ell} \log p = \delta_\ell \, (a_{\ell-1})^T$$

where $\delta_\ell = \partial \log p / \partial z_\ell \in \mathbb{R}^m$ is the backpropagated error at layer ℓ and $a_{\ell-1} \in \mathbb{R}^n$ is the incoming activation. Vectorising the outer product: $\mathrm{vec}(s_{W_\ell}) = a_{\ell-1} \otimes \delta_\ell$. Therefore:

$$G_\ell = E[\, \mathrm{vec}(s_{W_\ell})\, \mathrm{vec}(s_{W_\ell})^T \,] = E[\, (a_{\ell-1} \otimes \delta_\ell)(a_{\ell-1} \otimes \delta_\ell)^T \,] = E[\, (a_{\ell-1}\, a_{\ell-1}^T) \otimes (\delta_\ell\, \delta_\ell^T) \,]$$

If $a_{\ell-1}$ and δ_ℓ were statistically independent, the expectation would factor: $E[(aa^T) \otimes (\delta\delta^T)] = E[aa^T] \otimes E[\delta\delta^T]$. This is the K-FAC approximation:

Definition 4.1 (K-FAC Approximation)

The K-FAC approximation to the FIM block for layer ℓ is:

$$G_\ell \approx A_\ell \otimes S_\ell$$

where $A_\ell = E[a_{\ell-1}\, a_{\ell-1}^T] \in \mathbb{R}^{n \times n}$ is the covariance of input activations (the forward statistics) and $S_\ell = E[\delta_\ell\, \delta_\ell^T] \in \mathbb{R}^{m \times m}$ is the covariance of backpropagated gradients (the backward statistics). The inverse is then:

$$G_\ell^{-1} \approx A_\ell^{-1} \otimes S_\ell^{-1}$$

and the natural gradient for layer ℓ is computed as:

$$\tilde{g}_{W_\ell} = G_\ell^{-1}\, \nabla_{W_\ell} L \approx S_\ell^{-1}\, G_{W_\ell}\, A_\ell^{-1}$$

where $G_{W_\ell} = \nabla_{W_\ell} L$ is the ordinary gradient (an $m \times n$ matrix). The last line uses the identity $(A \otimes S)^{-1}\, \mathrm{vec}(G) = \mathrm{vec}(S^{-1} G A^{-1})$ to convert the P-dimensional vector-inverse problem into two small matrix inverses of sizes $n \times n$ and $m \times m$, each requiring $O(n^3)$ and $O(m^3)$ operations respectively.

Computational savings. For a layer with m outputs and n inputs, the exact FIM block G_ℓ has $mn \times mn$ entries ($O(m^2n^2)$ memory) and requires $O(m^3n^3)$ time to invert. K-FAC reduces this to $O(mn + m^2 + n^2)$ memory (for storing G, A, S) and $O(m^3 + n^3)$ time for the two small inversions. For typical layer sizes $m, n \in \{256, 512, 1024, 4096\}$, this represents a reduction from $O(10^{12})$ to $O(10^9)$ operations per layer — three orders of magnitude.

The independence assumption and its violations. The K-FAC approximation is exact when $a_{\ell-1}$ and δ_ℓ are statistically independent across training examples. This independence fails in practice because δ_ℓ depends on the

forward activations through the network's output distribution. However, empirical studies show that K-FAC provides accurate curvature estimates even though the independence assumption is violated: the Kronecker product captures the dominant structure of the FIM block, with the approximation error being relatively small and having limited impact on the direction of the natural gradient step.

4.5 EKFAC, Shampoo, and Modern Second-Order Methods

K-FAC has inspired several extensions and alternatives that address its limitations — primarily the independence assumption and the need for periodic inversion of the factor matrices.

EKFAC (Eigenvalue-corrected K-FAC). EKFAC (George et al., 2018) corrects for the approximation error in K-FAC by working in the basis of eigenvectors of A and S. If $A = U_A \Lambda_A U_A^T$ and $S = U_S \Lambda_S U_S^T$, then the K-FAC inverse is $(U_A \otimes U_S)(\Lambda_A^{-1} \otimes \Lambda_S^{-1})(U_A^T \otimes U_S^T)$. EKFAC replaces the Kronecker product $\Lambda_A^{-1} \otimes \Lambda_S^{-1}$ with a matrix Q^{-1} estimated empirically by projecting actual gradient outer products onto the eigenspace. This eigenvalue correction recovers some of the accuracy lost by the independence assumption without increasing the dominant $O(m^3 + n^3)$ computational cost.

Shampoo. Shampoo (Gupta et al., 2018; Anil et al., 2020) takes a different approach: instead of approximating the FIM, it maintains running estimates of the Kronecker factors of the gradient covariance matrix and inverts them using matrix powers. For a weight matrix $G \in \mathbb{R}^{m \times n}$, Shampoo maintains $L = \mathbb{E}[\Sigma_i G_i G_i^T] \in \mathbb{R}^{m \times m}$ and $R = \mathbb{E}[\Sigma_i G_i^T G_i] \in \mathbb{R}^{n \times n}$, and applies the preconditioned update $\theta_l \leftarrow \theta_l - \eta L^{-1/4} G R^{-1/4}$. Distributed Shampoo (Anil et al., 2022) parallelises the matrix inversion using eigendecompositions across TPU slices, achieving wall-clock times competitive with Adam while providing second-order convergence rates on language model pre-training benchmarks.

SOAP and AdaGrad-with-manifold. SOAP (Vyas et al., 2024) synthesises Shampoo's full preconditioning with Adam's momentum by maintaining the Shampoo eigenbasis and running Adam in that basis. The algorithm tracks the evolving eigenbasis of the gradient covariance and runs per-coordinate adaptive learning rates within this basis, providing a theoretically grounded synthesis of the two most successful optimisation paradigms. Empirically, SOAP matches or exceeds the best of Adam and Shampoo on language model pre-training when measured in steps, and approaches their wall-clock efficiency with careful implementation.

Method	Curvature Matrix	Storage (per layer)	Compute (per step)	Convergence
SGD	Identity	$O(P)$	$O(P)$	$O(1/\sqrt{T})$
Adam/AdamW	Diagonal FIM approx	$O(P)$	$O(P)$	$O(1/\sqrt{T})$ (adaptive)
K-FAC	Kronecker-block FIM	$O(m^2 + n^2)$ per layer	$O(m^3 + n^3)$ per T_inv	$O(1/T)$ near optimum
EKFAC	Eigenvalue-corrected Kronecker	$O(mn + m^2 + n^2)$	$O(m^3 + n^3)$ per T_inv	$O(1/T)$ with lower const
Shampoo	Full Kronecker factors	$O(m^2 + n^2)$	$O(m^3 + n^3)$	$O(1/T)$ provable
SOAP	Adam-in-Shampoo basis	$O(m^2 + n^2 + P)$	$O(m^3 + n^3 + P)$	Near Newton (empirical)

4.6 Practical Implementation and Hyperparameters

The gap between the theoretical appeal of natural gradient methods and their practical adoption is largely explained by implementation complexity and sensitivity to hyperparameters. This section addresses the key practical considerations for deploying K-FAC or its variants in large-scale training.

Damping. The FIM is positive semidefinite but may be nearly singular (small eigenvalues) due to overparameterisation, model symmetries (e.g., neuron permutation), and the fact that the empirical FIM is estimated on finite mini-batches. A damping term λ is added to regularise the inversion: $G_\ell \leftarrow G_\ell + \lambda I$, giving the update $G_\ell^{-1} \leftarrow (G_\ell + \lambda I)^{-1}$. Under the K-FAC approximation, this becomes $(A_\ell \otimes S_\ell + \lambda I)^{-1}$, which cannot be directly simplified to the pure Kronecker form. A common approximation is $(A_\ell + \sqrt{\lambda} I) \otimes (S_\ell + \sqrt{\lambda} I))^{-1}$, which maintains the Kronecker structure at the cost of solving a different regularised problem. The choice of λ is critical: too small and the inversion is numerically unstable; too large and the natural gradient degenerates to the Euclidean gradient.

Update frequency and amortised cost. Computing A_ℓ and S_ℓ from scratch requires one additional backward pass through layer ℓ, and inverting them requires $O(m^3 + n^3)$ operations. For large layers ($m, n \approx 4096$), this is comparable to a forward pass. K-FAC amortises this cost by updating the factors every T_{stats} steps (typically 10–50) and recomputing the inverses every T_{inv} steps (typically 100–500). Between updates, the cached inverse is applied at negligible additional cost. This amortised strategy reduces the per-step overhead of K-FAC to approximately 20–50% above that of Adam, while retaining most of the second-order benefit.

Learning rate and momentum. K-FAC is typically used with a learning rate 10–30× larger than Adam, because the natural gradient has already accounted for curvature and a step size of 1 corresponds to a unit step in the Fisher metric. Momentum is applied to the natural gradient rather than the Euclidean gradient: $m_t = \beta\, m_{t-1} + (1-\beta)\, \tilde{h}_t$, giving the update $\theta_{t+1} = \theta_t - \eta\, m_t$. Weight decay is applied directly to the parameters (as in AdamW) rather than to the gradient, for the same reasons discussed in Chapter 2.

The empirical performance of K-FAC is consistently superior to Adam in terms of convergence in steps (number of parameter updates), with typical speedups of 2–5× in step count to reach a given loss value. In wall-clock time, K-FAC is competitive with Adam for small to medium models but becomes more

expensive per step for very large layers (4096+ width), where the $O(n^3)$ inversion cost dominates. Distributed implementations on TPU clusters, where matrix inversions can be parallelised across cores, recover most of the wall-clock advantage. The ongoing development of memory-efficient second-order methods is one of the most active research areas in large-scale optimisation.

Conclusions: The Geometric Unity of Part I

The four chapters of this volume have traced a single thread through disparate mathematical territories: the geometry of weight matrices, the dynamics of gradient propagation, the curvature of distribution space, and the practical approximation of this curvature. Looking back, the unity of this thread becomes clear.

Linear algebra and the SVD provide the static geometry of neural network representations. The singular values of a weight matrix measure its effective rank, its approximation quality (Eckart-Young-Mirsky), and its degree of departure from a random Gaussian null model (Marchenko-Pastur). Low-rank structure — whether observed empirically in trained networks (LoRA), or exploited theoretically (tensor decompositions) — is the first and most basic geometric insight about why neural networks generalise despite their apparent overparameterisation.

Backpropagation, reframed as reverse-mode automatic differentiation, provides the dynamic geometry of gradient computation. The chain rule applied in reverse topological order to a computation graph is not merely an algorithm but a theorem: the gradient of a scalar loss can be computed in O(1) times the cost of a forward pass, regardless of the number of parameters. The vanishing gradient problem is a spectral stability condition on the product of layer Jacobians; residual connections solve it by adding an identity term to the product; normalisation layers address it by bounding the Frobenius norm of per-layer gradients uniformly in depth.

Information geometry provides the intrinsic geometry of the space of probability distributions parameterised by the network. The Fisher Information Matrix is the unique Riemannian metric that is invariant under reparameterisation, recovers the KL divergence at second order, and attains the Cramér-Rao lower bound as the precision of the maximum likelihood estimator. The natural gradient is steepest descent in this metric: provably reparameterisation-invariant, and equivalent to Newton's method applied to

the KL-divergence loss. Exponential families, dually flat manifolds, and the information bottleneck fill in the theoretical landscape connecting information geometry to the learning objectives used in practice.

K-FAC and its variants translate the theoretical insight of the natural gradient into practical second-order optimisers by exploiting the Kronecker product structure of the FIM under the assumption of independent activation and gradient statistics per layer. The resulting algorithms achieve near-Newton convergence rates in step count at a computational cost that is $O(n^3)$ per layer per inverse update — tractable for medium-sized networks and increasingly tractable for large ones as distributed implementations mature.

Together, these four pillars constitute the geometric foundations of deep learning: not merely mathematical prerequisites, but a coherent framework for understanding why the field's empirical successes are mathematically inevitable. Part II will extend this framework to the architecture level, asking what structural inductive biases are encoded in specific neural network designs and how group-theoretic and geometric constraints shape the space of functions a network can represent.

PART II

ARCHITECTURES AND THEIR MATHEMATICAL LOGIC

Inductive Biases and Symmetry Groups • The Transformer Family

Flash Attention, RoPE, Linear Attention, KV-Cache

Sequential Architectures: LSTM → HiPPO/S4 → Mamba/RWKV • GNN and the WL Test

Preface: On the Mathematical Logic of Architecture Design

Architecture design is not engineering intuition dressed in mathematical clothing. The choices that distinguish a Transformer from an LSTM, a graph neural network from a convolutional network, a selective state space model from a linear recurrent system — each of these choices encodes a precise mathematical prior about the structure of the data and the tasks the network will be asked to perform. This prior is called an inductive bias, and the mathematical language in which it is most precisely expressed is the language of group theory, symmetry, and representation theory.

This volume pursues that language with full rigour. We begin with the general theory of inductive biases and their algebraic formulation as equivariance constraints. We then apply this framework systematically to the Transformer family, tracing how each architectural innovation — scaled dot-product attention, multi-head attention, positional encodings, Flash Attention, RoPE, linear attention, KV-Cache — modifies the mathematical structure of the computation and its associated biases and efficiency properties. We then turn to sequential architectures: the LSTM (a learnable linear system with gated

nonlinearity), the HiPPO framework (optimal polynomial projection of input history), the S4 model (a linear recurrence with structured transitions), Mamba (content-selective SSMs), and RWKV (linear attention with decaying memory). Finally, we treat graph neural networks through the lens of the Weisfeiler-Lehman hierarchy of graph isomorphism tests, providing the sharpest known characterisation of their expressive power.

A recurring theme will be the tradeoff triangle: expressivity, computational efficiency, and sample efficiency. Every architectural choice moves a model along this triangle, and understanding the mathematical mechanism of each choice is the key to principled architecture design. We make this tradeoff explicit at every step.

Chapter 1: Inductive Biases, Symmetry Groups, and Equivariant Maps

An inductive bias is any constraint on the hypothesis class that is not derived from data. Every neural network architecture embodies inductive biases, whether or not they are consciously chosen. A fully connected network with unconstrained weight matrices has a relatively weak inductive bias — in principle it can represent any function, but this very generality means it must learn from scratch the regularity that domain-specific architectures build in. The question is not whether an architecture has inductive biases, but which biases it has and whether those biases align with the true regularity of the problem.

The most principled source of architectural inductive bias is symmetry. If the target function f: X → Y is known to respect a symmetry group G acting on X (and possibly Y), then restricting the hypothesis class to G-equivariant functions both reduces sample complexity (fewer parameters need to be learned) and improves generalisation (the learned function cannot violate the symmetry on new inputs). The mathematical formalisation of this idea is equivariance.

1.1 Group Actions and Equivariance

Let G be a group acting on spaces X and Y via group actions ϱ_X: G × X → X and ϱ_Y: G × Y → Y. A function f: X → Y is G-equivariant if:

$$f(\rho_X(g, x)) = \rho_Y(g, f(x)) \quad \text{for all } g \in G, x \in X$$

Intuitively: if we transform the input by g, the output transforms by g in the same way. The special case where ϱ_Y is the trivial action (g • y = y for all g) gives G-invariance: f(ϱ_X(g, x)) = f(x). Invariance is a stronger constraint than equivariance — invariant functions ignore the group action entirely, while equivariant functions track it in a structured way.

Examples of symmetry groups in deep learning.

- Translation group (ℝ^d, +): acting on images by pixel shifts. Convolution is translation-equivariant: if the image is shifted by t, the feature map shifts by t. This is the foundational bias of CNNs.
- Permutation group S_n: acting on sets of n elements by relabelling. A set-valued function (e.g., classifying a point cloud) should be permutation-invariant. PointNet achieves this via symmetric aggregation (max-pooling or sum-pooling over point features).
- Rotation group SO(3) and rigid motion group SE(3): acting on 3D molecular structures. Equivariant graph networks (e-GNN, SE(3)-Transformer, SEGNN) build in these symmetries for protein structure prediction and molecular property regression.
- Lorentz group SO(1,3): acting on 4-vectors in particle physics. LorentzNet builds exact Lorentz-equivariance for particle physics event classification.

Theorem 1.1 (Sample Complexity Reduction from Equivariance)

Let H be a hypothesis class of all continuous functions $X \to Y$ with Lipschitz constant L, and let $H_G \subset H$ be the subclass of G-equivariant functions. If G has measure $\mu(G)$ under a normalised Haar measure, then the Rademacher complexity satisfies:

$\mathfrak{R}_m(H_G) \leq \mathfrak{R}_m(H) / \sqrt{|G|}$ *(for finite G, $|G|$ = group order)*

This bound implies that building G-equivariance into the architecture is equivalent to augmenting the training set by a factor of $|G|$ in terms of its effect on generalisation bounds. For the permutation group S_n (order $n!$), this is an exponential reduction. For the translation group acting on $W \times H$ images (roughly WH translations), this corresponds to the well-known fact that data augmentation with translations is far less efficient than a convolutional architecture that builds translation equivariance in exactly.

1.2 Representation Theory and the Structure of Equivariant Linear Layers

The set of all G-equivariant linear maps between representation spaces is itself a vector space, and its structure is completely determined by the

representation theory of G. This is the key result that makes equivariant network design systematic rather than ad hoc.

Let ϱ: G → GL(V) and τ: G → GL(W) be linear representations of G on finite-dimensional vector spaces V and W over ℝ. A linear map T: V → W is equivariant (an intertwiner) if Tϱ(g) = τ(g)T for all g ∈ G. By Schur's Lemma:

Theorem 1.2 (Schur's Lemma — Structure of Equivariant Maps)

(i) If ρ and τ are irreducible representations (irreps) over ↛ then every equivariant map T: V ↛ W is either zero (if ρ ≉ τ) or a scalar multiple of the canonical isomorphism (if ρ ≅ τ).

(ii) More generally, by the decomposition of ρ and τ into irreps ρ = ⊕_k m_k ρ_k and τ = ⊕_k n_k ρ_k, the space of equivariant maps decomposes as:

Hom_G(V, W) ≅ ⊕_k 𝕽^{n_k × m_k}

so the number of free parameters in an equivariant linear layer is Σ_k n_k m_k, compared to (dim V) (dim W) for an unconstrained linear map. The multiplicities m_k, n_k count how many times each irrep appears in the input and output representations. This decomposition is the mathematical basis for equivariant network design: represent the data as a direct sum of irreps, and parametrise layers as block-diagonal in the irrep basis with learnable multiplicity-mixing matrices.

Steerable CNNs. Cohen and Welling's Steerable CNNs implement this programme for the 2D rotation group SO(2) and the dihedral groups D_n. The feature maps at each layer are G-steereable vector bundles: at each spatial location, the feature vector transforms as a direct sum of irreducible representations of the local symmetry group. The learnable filter banks are constrained to be equivariant (their weights lie in the spaces Hom_G(V_in, V_out) determined by Schur's Lemma), and the spatial convolution provides the global translation equivariance. The result is a network that is exactly equivariant to the full group G = ℝ^2 ⋊ SO(2) of planar Euclidean motions.

1.3 The Universal Approximation Theorem for Equivariant Functions

A natural question is whether imposing equivariance constraints limits the class of functions a network can approximate. The answer is negative: the space of continuous G-equivariant functions is dense in the space of all G-equivariant functions under mild conditions, and equivariant networks are universal approximators within this restricted but natural class.

Theorem 1.3 (Equivariant Universal Approximation)

Let G be a compact group acting continuously on compact Hausdorff spaces X and Y. Let H_G be the space of continuous G-equivariant functions X → Y with the uniform norm. Then the class of equivariant neural networks with sufficient width and depth is dense in H_G.

The key steps in the proof are: (1) The equivariant Peter-Weyl theorem decomposes H_G into irreducible components indexed by the dual Ĝ of G; (2) each irreducible component is approximated by equivariant polynomials (via the Stone-Weierstrass theorem for G-algebras); (3) equivariant polynomials are expressed as products of equivariant linear maps and pointwise nonlinearities. The theorem implies that no expressivity is lost by imposing equivariance: equivariant networks can represent any continuous equivariant function, and only continuous equivariant functions.

Chapter 2: The Transformer Family

The Transformer architecture (Vaswani et al., 2017) is the dominant architecture in contemporary deep learning. Its mathematical core — scaled dot-product attention — is a kernel method: it computes a weighted combination of value vectors, where weights are determined by a kernel function (the dot product) evaluated between query and key vectors. Understanding the Transformer at this level of mathematical depth reveals why it is expressive, why it scales, and why each subsequent innovation (Flash Attention, RoPE, linear attention, KV-Cache) addresses a specific mathematical limitation of the original design.

2.1 Scaled Dot-Product Attention: Exact Formulation

Let X ∈ ℝ^{L×d} be a sequence of L token embeddings of dimension d. For a single attention head, the queries, keys, and values are obtained by linear projections:

$$Q = X\,W_Q \in \mathbb{R}^{L\times d_k},\quad K = X\,W_K \in \mathbb{R}^{L\times d_k},\quad V = X\,W_V \in \mathbb{R}^{L\times d_v}$$

where W_Q, W_K ∈ ℝ^{d×d_k} and W_V ∈ ℝ^{d×d_v} are the projection matrices. The attention output is:

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}(QK^T / \sqrt{d_k})\,V \in \mathbb{R}^{L\times d_v}$$

The attention weight matrix A = softmax(QK^T / √d_k) ∈ ℝ^{L×L} has entry A_{ij} = exp(q_i^T k_j / √d_k) / Σ$_k$ exp(q_i^T k_$_k$ / √d_k), representing the normalised affinity between position i (as a query) and position j (as a key). The output at position i is the A_{ij}-weighted average of value vectors: out_i = Σ_j A_{ij} v_j.

The √d_k scaling. If q_i and k_j are independently sampled from ℕ(0, I_{d_k}), their dot product q_i^T k_j has variance d_k. Without scaling, the pre-softmax logits grow as √d_k in standard deviation, pushing the softmax into saturation (near-zero gradients everywhere except at the argmax). Dividing by √d_k normalises the variance to 1, keeping the softmax in its gradient-informative

regime throughout training. This scaling is an instance of a general principle: the input to a softmax should be O(1) in magnitude for well-conditioned gradients.

Multi-head attention. Multi-head attention runs h parallel attention heads with independent projections (W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}) and concatenates their outputs:

$$MHA(X) = Concat(head_1, ..., head_h)\, W_O, \quad head_i = Attn(X\, W_Q^{(i)}, X\, W_K^{(i)}, X\, W_V^{(i)})$$

where W_O ∈ ℝ^{hd_v × d} projects the concatenated heads back to the model dimension. Typically d_k = d_v = d/h so the total parameter count is O(d^2) per attention layer regardless of the number of heads. Each head can specialise in a different type of similarity (syntactic, semantic, positional), and the multi-head structure provides an ensemble of kernel machines operating simultaneously.

Theorem 2.1 (Attention as a Universal Approximator of Sequence-to-Sequence Maps)

A single-layer Transformer with sufficient number of attention heads and embedding dimension can approximate any continuous sequence-to-sequence function f: (ℝ^d)^L → (ℝ^d)^L on compact domains to arbitrary precision. The proof proceeds by showing that multi-head attention can implement any weighted sum over the input sequence, and that the feed-forward sublayer can then apply an arbitrary continuous function to each position independently. Combining the two sublayers, any pointwise-continuous permutation-equivariant function is reachable. With positional encodings breaking the permutation equivariance, any continuous function on fixed-length sequences is approximable.

Depth (multiple layers) is needed for functions that require multi-round reasoning: each layer can refine the representation based on the globally attended context from the previous layer. Empirically, depth is more parameter-efficient than width for complex compositional tasks.

2.2 Positional Encodings: From Additive Sinusoids to RoPE

The attention mechanism as defined above is permutation-equivariant: any permutation of the input sequence produces a permuted output. This is problematic for tasks where order matters (language modelling, time series

forecasting, machine translation). Positional encodings break this symmetry by injecting position-dependent information into the input representations.

Sinusoidal positional encodings (Vaswani et al., 2017). The original Transformer adds to each input embedding x_t a fixed positional vector PE(t) with components:

$$PE(t, 2k) = sin(t / 10000^{2k/d}), \quad PE(t, 2k+1) = cos(t / 10000^{2k/d})$$

The choice of sinusoids is motivated by the observation that PE(t + Δt) is a linear function of PE(t): there exists a rotation matrix R_{Δt} such that PE(t + Δt) = R_{Δt} PE(t). This means the attention mechanism can in principle learn to attend to tokens at a fixed relative offset Δt by learning to apply R_{Δt} to the queries or keys. However, the linearity holds for the raw positional vectors, not for the projected queries and keys — the projections W_Q and W_K distort the linear relationship, making relative position information harder to exploit than the design intention suggests.

Rotary Position Embedding (RoPE, Su et al., 2021). RoPE addresses this by encoding position directly in the dot product between query and key, rather than in the input embeddings. The key insight is to rotate the query and key vectors for position t by a rotation matrix Φ_t:

$$\alpha(q_t, k_s) = (R_t\, q_t)^T (R_s\, k_s) = q_t^T R_t^T R_s\, k_s = q_t^T R_{s-t}\, k_s$$

where R_t is a block-diagonal rotation matrix with 2×2 rotation blocks for each pair of dimensions, rotated by angle tθ_k = t × 10000^{-2k/d} for the k-th dimension pair. The resulting attention score depends only on the relative position s - t, not on the absolute positions s and t separately. This is the relative position equivariance property that the original sinusoidal encoding could not achieve exactly.

Theorem 2.2 (RoPE Relative Position Equivariance)

Let $R_t \in SO(d)$ be the RoPE rotation at position t, defined as the block-diagonal matrix with 2×2 rotation blocks $R(\theta_k t)$ for k = 1,...,d/2. Then for any query q and key k:

$(R_t\, q)^T (R_s\, k) = q^T R_{s-t}\, k$

The proof follows from the identity R_t^T R_s = R_{s-t} (rotation by -(tθ_k) followed by rotation by sθ_k equals rotation by (s-t)θ_k for each block). This means the dot-product attention score with RoPE depends only on the relative displacement s-t between positions, making the attention pattern translation-equivariant. RoPE also extrapolates better than additive sinusoidal encodings to sequence lengths beyond those seen during training because the relative rotation angle (s-t)θ_k grows linearly in the displacement, maintaining consistent angular scale at any sequence length.

ALiBi and other positional schemes. ALiBi (Press et al., 2022) takes a simpler approach: instead of modifying the queries and keys, it adds a head-specific linear bias -m_h |s - t| to the pre-softmax attention score at head h, where m_h is a head-specific slope. This penalises long-range attention linearly in distance, implementing a soft local attention bias. ALiBi is parameter-free (m_h is fixed, not learned), has excellent length extrapolation properties (the linear penalty smoothly extends to unseen distances), and has become the positional encoding of choice for several production language models (BLOOM, MPT). Its simplicity, however, comes at the cost of expressivity: it cannot represent arbitrary relative position functions, only linear-decay functions.

2.3 Flash Attention: IO-Aware Exact Attention

The standard attention computation requires materialising the L×L attention weight matrix A, which costs O(L^2) memory. For L = 100,000 (typical for long-context models), this is 10^{10} entries — 40 GB in float32. Flash Attention (Dao et al., 2022; Dao, 2023) reduces this to O(L) memory while computing exactly the same output as standard attention, by exploiting the hierarchical memory structure of GPU hardware.

The memory bottleneck. Modern GPUs have a fast on-chip SRAM (typically 20–40 MB for an A100) and a large off-chip HBM (40–320 GB). The bandwidth to SRAM is ∼19 TB/s but its capacity is tiny; the bandwidth to HBM is ∼2 TB/s. Standard attention requires multiple HBM reads: read Q, K to compute QK^T (write to HBM); read QK^T, apply softmax (write to HBM); read A, V to

compute AV (write output). Each read/write is an HBM access, and HBM bandwidth is the bottleneck.

Tiled computation. Flash Attention processes the sequence in tiles of size B_r × B_c (block rows × block cols), where B_r and B_c are chosen so that tiles of Q, K, V fit in SRAM simultaneously. For each tile of queries (rows), it iterates over all tiles of keys and values (cols), accumulating the softmax-weighted value sum in SRAM without ever writing the full L×L attention matrix to HBM. The softmax normalisation across tiles requires the online softmax algorithm:

Algorithm 2.1 (Online Softmax for Flash Attention)

To compute softmax(x_1, ..., x_L) without storing all x_i simultaneously, maintain running statistics (m, d) where m = max seen so far and d = sum of exp(x_i - m) seen so far:

For each new x_{i+1}: m_new = max(m, x_{i+1}); d_new = d × exp(m - m_new) + exp(x_{i+1} - m_new)

The correction factor exp(m - m_new) rescales the previous sum to the new max. At the end, softmax_i = exp(x_i - m) / d. This algorithm is numerically stable (equivalent to standard softmax after subtracting the global max) and requires O(1) memory per streaming pass.

Flash Attention extends this to matrix blocks: for a query block Q_i and key-value blocks (K_j, V_j), maintain per-row running max m_i^{(j)} and sum d_i^{(j)}, and accumulate the partial attention output O_i^{(j)} = A_{ij}V_j (where A_{ij} = exp(Q_i K_j^T/√d_k - m_i^{(j)})). After processing all key blocks, the final output O_i = Σ_j O_i^{(j)} × (d_i^{(j)}/d_i^{(final)}) is exact.

Complexity analysis. Standard attention: O(L² d) FLOPs, O(L² + Ld) memory, O(L²/B) HBM reads. Flash Attention: O(L² d) FLOPs (same!), O(Ld) memory (reduced from O(L²)), O(L² d / M) HBM reads (where M is SRAM size). The FLOP count is unchanged — Flash Attention does not reduce the arithmetic work. Its advantage is entirely in HBM bandwidth, which is the true bottleneck for L ≥ 1024 on modern GPUs. Empirically, Flash Attention achieves 2–4× wall-clock speedup over standard attention at L = 2048 and 5–10× at L = 4096 on A100 GPUs.

Version	Memory	FLOPs	Backward Pass	Key Feature

Standard Attention	$O(L^2)$	$O(L^2 d)$	Standard autodiff	Simple but memory-bound
Flash Attention v1	$O(Ld)$	$O(L^2 d)$	Recompute A in backward	2-4× speedup at L=2048
Flash Attention v2	$O(Ld)$	$O(L^2 d)$	Optimised backward	Better GPU utilisation
Flash Attention v3	$O(Ld)$	$O(L^2 d)$	Async overlapped	Optimised for H100/Hopper
Flash Decoding	$O(Ld)$	$O(Ld)$	N/A (inference)	Parallelises KV-Cache lookup

2.4 KV-Cache: The Memory-Bandwidth Tradeoff at Inference

At inference time, transformer language models generate tokens autoregressively: given tokens (t_1, ..., t_n), they predict t_{n+1}, then given (t_1, ..., t_{n+1}) they predict t_{n+2}, and so on. Naively, each forward pass recomputes the keys and values for all previous tokens, costing $O(nL)$ per token (n is the current position, L the context length). The KV-Cache eliminates this redundancy by caching the computed key and value matrices from previous steps.

Mechanism. At step n, the keys and values for the new token t_n are computed and appended to the cache: K_cache ← [K_cache; k_n^T], V_cache ← [V_cache; v_n^T]. The attention at step n then reads: Q_n = q_n (1×d_k), K_cache (n×d_k), V_cache (n×d_v), and computes the attention output using the cached KV without recomputation. The cost per token drops from $O(n^2 d)$ (naive) to $O(nd)$ (with cache) — a factor of n reduction in compute.

Memory cost. The KV-Cache stores 2 × n_layers × n_heads × n × d_head values per sequence in the batch. For a typical large model (96 layers, 96 heads, d_head = 128), with n = 100,000 tokens and fp16 precision, this is 2 × 96 × 96 × 100,000 × 128 × 2 bytes ≈ 470 GB per sequence — exceeding the GPU memory of

any current hardware. This is the primary challenge of long-context inference and motivates a family of KV-Cache compression techniques:

- Quantisation: store KV in 4-bit or 8-bit formats (KVQuant, KIVI), accepting small approximation error for 4-8× memory reduction.
- Eviction: dynamically discard cache entries for low-attention-weight tokens (H2O, ScissorHands), keeping only the most attended tokens.
- Sparse attention: at inference, compute attention only over a sliding window or over selected anchor tokens (Sliding Window, StreamingLLM).
- Multi-Query Attention (MQA): share a single key-value head across all query heads. Cache size drops by factor h (number of heads). Used in Falcon, PaLM-2.
- Grouped-Query Attention (GQA): intermediate between MHA and MQA. Group query heads into G groups, with one KV head per group. Cache drops by factor h/G. Used in Llama-2 70B, Mistral, Gemma.

Theorem 2.3 (GQA Approximation Quality)

Let MHA be multi-head attention with h heads and GQA with G < h groups (h/G queries per group). GQA is equivalent to MHA with the constraint that all queries in group g share the same key and value projections W_K^{(g)}, W_V^{(g)}. The approximation error of GQA relative to MHA decreases as G increases: G = h recovers MHA exactly, G = 1 gives MQA. For any fixed target accuracy ε > 0 in terms of output Frobenius norm, there exists G = G*(h, ε, X) such that GQA with G ≥ G* groups achieves output error ≤ ε for input X. Empirically, G = h/8 (e.g., G = 8 for h = 64) recovers essentially all of MHA's performance at 8× reduced KV-Cache size.*

2.5 Linear Attention and the Kernel Interpretation

Standard softmax attention has $O(L^2)$ cost due to the L×L attention matrix. Linear attention replaces the softmax with a different, factorised attention

kernel that decomposes the L×L matrix into a product of L×d and d×L matrices, reducing cost to $O(Ld^2)$.

The key observation is that softmax attention can be written as:

$$out_i = \Sigma_j \exp(q_i^T k_j / \sqrt{d_k})\, v_j \,/\, \Sigma_j \exp(q_i^T k_j / \sqrt{d_k}) = \Sigma_j K(q_i, k_j)\, v_j \,/\, \Sigma_j K(q_i, k_j)$$

where $K(q, k) = \exp(q^T k / \sqrt{d_k})$ is a positive definite kernel. Linear attention replaces K with a kernel that factors: $K(q, k) = \varphi(q)^T \varphi(k)$ for some feature map $\varphi: \mathbb{R}^{d_k} \to \mathbb{R}^r$. Then:

$$out_i = \varphi(q_i)^T (\Sigma_j \varphi(k_j)\, v_j^T) \,/\, \varphi(q_i)^T (\Sigma_j \varphi(k_j))$$

The sum $S = \Sigma_j \varphi(k_j)\, v_j^T \in \mathbb{R}^{r \times d_v}$ and $z = \Sigma_j \varphi(k_j) \in \mathbb{R}^r$ can be computed incrementally: processing $j = 1, \ldots, L$ sequentially costs $O(Lrd_v)$ total. If $r = O(d_k)$, this is $O(Ld_k d_v) = O(Ld^2/h^2)$ — linear in L.

Feature maps and the exponential kernel. The challenge is choosing φ such that $\varphi(q)^T \varphi(k)$ approximates $\exp(q^T k / \sqrt{d_k})$ well while keeping r small. Random Fourier Features (Rahimi and Recht, 2007) approximate shift-invariant kernels by sampling Fourier components. For the exponential kernel, exact positive feature maps exist: Performer (Choromanski et al., 2021) uses orthogonal random features FAVOR+ based on:

$$\varphi(x) = \exp(-\|x\|^2/2) \times (\exp(\omega_1^T x), \ldots, \exp(\omega_r^T x)) \quad \omega_i \sim \mathcal{N}(0, I/\sqrt{d_k})$$

Retentive Networks and the recurrence-attention duality. RetNet (Sun et al., 2023) and related models (GLA, HGRN, RWKV) exploit the fact that linear attention with causal masking is equivalent to a linear recurrent neural network. Specifically, for causal linear attention with decay γ:

$$s_t = \gamma\, s_{t-1} + \varphi(k_t)\, v_t^T, \quad out_t = \varphi(q_t)^T s_t \,/\, normaliser$$

The hidden state $s_t \in \mathbb{R}^{r \times d_v}$ is the compressed summary of all past keys and values, updated by a simple outer product addition with exponential decay. During training, this can be parallelised using the chunk-wise form (compute attention within chunks in parallel, pass hidden states between chunks sequentially). During inference, it reduces to an O(1)-per-token

recurrence. This training-inference duality is the core advantage of linear attention over quadratic attention and of recurrent over attention-based models.

Chapter 3: Sequential Architectures — From LSTM to Mamba

The problem of learning from sequential data with long-range dependencies is one of the oldest challenges in machine learning. The progression from Hidden Markov Models through LSTMs to state space models is a mathematical story of increasingly powerful representations of memory, each with characteristic tradeoffs between the length of dependency, the cost of training, and the cost of inference. The past three years have produced a genuine architectural revolution in this space: selective state space models (Mamba) and hybrid architectures challenge the transformer's dominance on sequence modelling tasks while offering O(L) inference cost rather than $O(L^2)$.

3.1 Long Short-Term Memory: The Gating Solution to Vanishing Gradients

The LSTM (Hochreiter and Schmidhuber, 1997) was designed to solve the vanishing gradient problem for recurrent networks by introducing an explicit memory cell c_t with additive (rather than multiplicative) updates. Its gating mechanism is the archetype of all subsequent gated architectures.

The LSTM equations for hidden state h_{t-1} ∈ ℝ^h and cell state c_{t-1} ∈ ℝ^h, given input x_t ∈ ℝ^d:

$$i_t = \sigma(W_i\,[h_{t-1};\, x_t] + b_i) \quad \textit{[input gate]}$$

$$f_t = \sigma(W_f\,[h_{t-1};\, x_t] + b_f) \quad \textit{[forget gate]}$$

$$o_t = \sigma(W_o\,[h_{t-1};\, x_t] + b_o) \quad \textit{[output gate]}$$

$$g_t = \tanh(W_g\,[h_{t-1};\, x_t] + b_g) \quad \textit{[cell candidate]}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh(c_t)$$

The cell state update c_t = f_t ⊙c_{t-1} + i_t ⊙g_t is additive when f_t ≈ 1 and i_t ≈ 0 (memorising), allowing gradients to flow back through many time steps

without decay. The forget gate f_t controls how much of the previous cell state is retained; if f_t ≈ 1 for many steps, the gradient ∂c_t/∂c_{t-k} = ∏τ f_τ ≈ 1, preventing vanishing. This is mathematically identical to the residual connection argument in Chapter 2 of Part I: an additive update prevents the product of Jacobians from decaying exponentially.

Theorem 3.1 (LSTM Gradient Flow)

For an LSTM, the gradient of the loss L at time T with respect to the cell state c_t is:

∂L/∂c_t = (∂L/∂c_T) × ∏_{τ=t+1}^{T} f_τ + lower-order terms

If the forget gates f_τ ∈ (0, 1) for all τ, then the gradient decays as ∏ f_τ < 1 but polynomially (not exponentially) in (T-t) if f_τ is bounded away from 0. If f_τ = 1 for all τ (open memory gate), the gradient is exactly preserved. This is in contrast to the vanilla RNN where the analogous product is ∏τ σ′(z_τ) ρ(W_h), which decays exponentially when the spectral radius ρ(W_h) × ‖σ′‖∞ < 1.

3.2 The HiPPO Framework: Optimal Polynomial Memory

The HiPPO (High-order Polynomial Projection Operator) framework (Gu et al., 2020) provides a mathematically principled approach to the question: what is the optimal way for a recurrent system to remember its input history? The answer is to project the input history onto an orthogonal polynomial basis, and HiPPO derives the exact linear recurrence that implements this projection optimally.

Formal setup. Let u: [0, t] → ℝ be the input signal up to time t, and let ω(x, t) be a time-varying measure on [0, t] (e.g., uniform: ω(x, t) = 1/t). HiPPO seeks coefficients c(t) ∈ ℝ^N such that the N-dimensional polynomial approximation g(x, t) = Σ_{n=0}^{N-1} c_n(t) P_n(x) minimises the weighted L^2 error:

$$c(t) = \operatorname{argmin}_{c \in \mathbb{R}^N} \int_0^t (u(x) - g(x, t))^2 \, \omega(x, t) \, dx$$

HiPPO derives that for any such measure ω, the optimal coefficients satisfy a linear ODE:

$$c'(t) = A(t)\, c(t) + B(t)\, u(t)$$

where A(t) and B(t) are explicitly computable from the measure ω and the chosen polynomial basis. This is a continuous-time linear recurrence, and it is the theoretically grounded generalisation of the LSTM cell state update to an arbitrary order of polynomial approximation.

Theorem 3.2 (HiPPO-LegS: Legendre Scaling Measure)

For the scaled Legendre measure ω(x, t) = 1/t × 𝟙{[0,t]}(x) (uniform over all history up to t), the HiPPO matrix A ∈ ℝ^{N×N} has entries:

A_{nk} = -(2n+1)^{1/2}(2k+1)^{1/2} if n > k

A_{nk} = -(n+1) if n = k

A_{nk} = 0 if n < k

and B_n = (2n+1)^{1/2}. The coefficients c_n(t) are the Legendre polynomial coefficients of the input history u(s), s ≤ t, under the uniform measure. This means the HiPPO state c(t) ∈ ℝ^N encodes the best N-th order polynomial approximation to the entire input history, with an approximation error that decays as O(1/N^2) in L². Critically, the A matrix is lower-triangular, making the recurrence causal and stable.

The significance of HiPPO is that it provides a principled justification for the specific structure of the recurrence matrices in S4 and its descendants. Rather than treating the recurrence matrix A as a free parameter to be learned from data (as in vanilla RNNs and LSTMs), HiPPO derives A analytically from the desired memory property. The result is a recurrence that can provably memorise input history of arbitrary length, limited only by the number of polynomial basis functions N.

3.3 S4: Structured State Spaces for Sequences

The S4 model (Gu et al., 2021) instantiates the HiPPO framework as a trainable neural network layer. A continuous-time state space model (SSM) is defined by the system:

$$x'(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t) + Du(t)$$

where x(t) ∈ ℝ^N is the hidden state, u(t) ∈ ℝ is the scalar input, y(t) ∈ ℝ is the output, and A ∈ ℝ^{N×N}, B ∈ ℝ^{N×1}, C ∈ ℝ^{1×N}, D ∈ ℝ are the system matrices. Discretising with step size Δ using the bilinear (Tustin) method:

$$\bar{A} = (I - \Delta/2 \cdot A)^{-1} (I + \Delta/2 \cdot A), \quad \bar{\mathcal{B}} = (I - \Delta/2 \cdot A)^{-1} \Delta B$$

gives the discrete recurrence x_k = Ā x_{k-1} + B̄ u_k, y_k = C x_k + D u_k. Unrolling this recurrence gives a convolution y = K * u where K_k = C Ā^{k-1} B̄ is the SSM convolution kernel (also called the impulse response). Computing y via convolution costs O(L log L) using FFT, compared to O(L^2) for naive recurrence unrolling.

Theorem 3.3 (SSM Convolution and Diagonalisation)

If A is diagonalisable as $A = V \Lambda V^{-1}$ with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$, then:

$K_k = C \bar{A}^{k-1} \bar{\mathcal{B}} = (CV)\, \Delta\, \bar{A}^{k-1}\, (V^{-1}B) = \sum_{n=1}^{N} \alpha_n \bar{\lambda}_n^{k-1}$

where $\alpha_n = (CV)_n (V^{-1}B)_n$ and $\bar{\lambda}_n$ is the n-th eigenvalue of $\bar{A}$ (a Möbius transform of λ_n). The kernel K decomposes as a sum of N exponential sequences, one per eigenvalue. For stability, $|\bar{\lambda}_n| < 1$ (equivalently, $\mathrm{Re}(\lambda_n) < 0$). The diagonalisation reduces the N×N matrix power $\bar{A}^k$ to N scalar powers $\bar{\lambda}_n^k$, making the kernel computation O(NL) rather than $O(N^2 L)$.

S4 uses the NPLR (Normal Plus Low-Rank) structure: $A = \Lambda - PQ^T$ where Λ is diagonal and $P, Q \in \mathbb{R}^{N \times r}$ with $r \ll N$. The HiPPO-LegS matrix has exactly this structure with r = 1. This structure enables efficient computation of the resolvent $(sI - A)^{-1}$ using the Woodbury identity, which is the key operation in the frequency-domain computation of the SSM kernel.

3.4 Mamba: Selective State Space Models

S4 and its immediate successors (S4D, H3, Hyena) are linear time-invariant (LTI) systems: the matrices A, B, C do not depend on the input u. This is mathematically clean and enables the O(L log L) convolution computation, but it is also a fundamental limitation: the system cannot selectively focus on or ignore specific inputs based on their content. Mamba (Gu and Dao, 2023) addresses this by making B, C (and optionally Δ) functions of the input, creating a selective SSM.

Selectivity mechanism. In Mamba, the discretisation step Δ and the input projection B and output projection C are computed from the current input u_t via learned linear projections:

$$\Delta_t = softplus(W_\Delta\, u_t + b_\Delta) \in \mathbb{R}^d$$

$$B_t = W_B\, u_t \in \mathbb{R}^N, \quad C_t = W_C\, u_t \in \mathbb{R}^N$$

This input-dependence means that the state update x_t = $\bar{A}$(Δ_t) x_{t-1} + $\bar{B}$(Δ_t, B_t) u_t can selectively remember or forget based on the content of u_t. Large Δ_t $\to \infty$ makes $\bar{A} \to 0$, I (the state is reset or accumulated depending on B_t); small Δ_t $\to 0$ makes $\bar{A} \to$ I (the state is preserved unchanged). The model can thus learn to ignore irrelevant tokens (small Δ_t, reset), accumulate relevant ones (large Δ_t with appropriate B_t), and selectively retrieve with learned C_t.

Theorem 3.4 (Mamba Selectivity and the Limitation of Linear Attention)

A linear time-invariant SSM with fixed A, B, C cannot solve the Selective Copying task: given a sequence of inputs u_1, ..., u_L where a subset are marked as 'select', output those selected inputs in order. This task requires the system to condition its state update on the content of the input (whether it is marked), which is by definition not possible for a fixed-coefficient linear system.

Mamba's input-dependent Δ_t, B_t, C_t resolve this: a large Δ_t for marked inputs causes them to be written into the state (via B_t), while unselected inputs with small Δ_t leave the state unchanged. Selective Copying, Induction Heads, and similar tasks that require content-dependent memory are solved by Mamba but not by S4/Hyena. This theoretical gap corresponds to the empirical finding that Mamba matches or surpasses transformers on discrete sequence modelling tasks (language, DNA, time series) while using O(L) rather than $O(L^2)$ memory.

Parallel scan for training. Mamba's input-dependent recurrence cannot be computed as a convolution (the kernel changes at every step). Naive sequential computation costs O(L) steps, unparallelisable. The key algorithmic innovation is the parallel prefix scan (also called scan or prefix sum), which computes the recurrence x_t = a_t x_{t-1} + b_t for input-dependent a_t, b_t in O(log L) parallel steps using a balanced binary tree reduction:

$$Scan(a, b): O_t = \prod_{\tau=1}^{t} a_\tau \times x_0 + \Sigma_{\tau=1}^{t} (\prod_{s=\tau+1}^{t} a_s)\, b_\tau$$

The parallel scan is implemented as a custom CUDA kernel that performs the tree reduction in shared memory, achieving O(L) total work (optimal) in O(log L) parallel steps. This, combined with a hardware-efficient memory management strategy similar to Flash Attention (recomputing activations in the backward pass from the compressed recurrent state), gives Mamba a practical training cost of O(Ld) — similar to a transformer with $O(L^2d)$ attention replaced by O(Ld) selective scanning.

3.5 RWKV: Recurrence with Linear Attention

RWKV (Peng et al., 2023) takes a different approach to efficient sequence modelling: it reformulates linear attention as a recurrence, enabling the model to run as an RNN at inference while training in parallel as an attention model. The name RWKV stands for the four model components: Receptance (R), Weight (W), Key (K), Value (V).

The WKV operator. At time t, RWKV computes the attention-like output:

$$wkv_t = (\Sigma_{s<t} \exp(-(t-1-s)w + k_s)\, v_s + \exp(u + k_t)\, v_t) \,/\, (\Sigma_{s<t} \exp(-(t-1-s)w + k_s) + \exp(u + k_t))$$

where $w \in \mathbb{R}^d$ is a learned per-channel time decay vector (always negative so $\exp(-s \cdot w)$ decays with time), $u \in \mathbb{R}^d$ is a learned per-channel bonus for the current token, and k_t, v_t are the key and value at time t. This can be rewritten as a recurrence by defining:

$$a_t = \exp(-w) \odot a_{t-1} + \exp(k_t) \odot v_t, \quad b_t = \exp(-w) \odot b_{t-1} + \exp(k_t)$$

$$wkv_t = (a_{t-1} + \exp(u + k_t)\, v_t) \,/\, (b_{t-1} + \exp(u + k_t))$$

The recurrence is linear in (a_t, b_t), making it computable in O(1) per step at inference and parallelisable in O(L) via parallel prefix scan during training. RWKV-4 (the first production-scale version) achieved performance competitive with GPT-3 class transformers on language modelling benchmarks while using constant memory at inference, enabling deployment in memory-constrained environments.

Architecture	Training Cost	Inference Cost (per token)	Memory (inference)	Long-range Dependencies
Transformer	$O(L^2d)$	$O(Ld)$ with KV-Cache	$O(L)$ cache	Full $O(L)$ receptive field
LSTM	$O(Ld)$	$O(d^2)$ matrix multiply	$O(d)$	Exponential decay
S4	$O(L \log L \cdot d)$	$O(Nd)$	$O(N)$	Polynomial decay (HiPPO)
Mamba	$O(Ld)$ parallel scan	$O(Nd)$	$O(N)$	Selective: learned decay
RWKV	$O(Ld)$ prefix scan	$O(d)$	$O(d)$	Exponential decay (learnable w)
Hybrid (Jamba)	$O(L^2d) + O(Ld)$	$O(d) + O(Ld)$ hybrid	Reduced cache	Both: local + global

Chapter 4: Graph Neural Networks and the Weisfeiler-Leman Hierarchy

Graph-structured data is pervasive in science and technology: molecular graphs in drug discovery, social networks in recommendation systems, knowledge graphs in question answering, scene graphs in computer vision, and mesh graphs in physics simulation. Graph Neural Networks (GNNs) are the dominant framework for learning on such data. Unlike sequence or image data, graphs have no canonical ordering of nodes — any relabelling of nodes that preserves graph structure should produce equivalent outputs. This permutation symmetry is the fundamental inductive bias of GNNs, and the Weisfeiler-Leman (WL) graph isomorphism test provides the sharpest theoretical characterisation of how well GNNs can exploit it.

4.1 Message-Passing Neural Networks: The General Framework

A graph G = (V, E) consists of nodes V = {1, ..., n} with node features h_v^{(0)} ∈ ℝ^d and (optionally) edge features e_{uv} ∈ ℝ^{d_e} for edges (u, v) ∈ E. A Message-Passing Neural Network (MPNN, Gilmer et al., 2017) applies L rounds of neighbourhood aggregation:

$$m_v^{(t)} = AGG_t(\{MSG_t(h_v^{(t-1)}, h_u^{(t-1)}, e_{uv}) : u \in \mathcal{N}(v)\})$$

$$h_v^{(t)} = UPD_t(h_v^{(t-1)}, m_v^{(t)})$$

where AGG is a permutation-invariant aggregation function (sum, mean, max, attention-weighted sum), MSG is a message function, and UPD is an update function. After L rounds, a readout function produces the graph-level prediction: ŷ = READOUT({h_v^{(L)} : v ∈ V}). Specific choices of MSG, AGG, UPD recover all major GNN architectures: GCN (fixed normalised mean aggregation), GraphSAGE (learnable mean), GAT (attention-weighted), GIN (injective MLP aggregation), MPNN (explicit edge features).

Permutation equivariance. The MPNN framework is permutation equivariant by construction: reordering the nodes of G produces a reordering of the node representations {h_v^{(L)}}, and the graph-level readout (a permutation-invariant function of node representations) is permutation invariant. This is the correct inductive bias for graph-structured data: the graph's properties do not depend on how we label its nodes.

4.2 The Weisfeiler-Leman Test: Graph Isomorphism and Its Hierarchy

The Weisfeiler-Leman (WL) graph isomorphism test is a classical algorithm for determining whether two graphs are isomorphic. It provides the theoretical ceiling on the expressive power of message-passing GNNs. Understanding the WL test and its generalisations is the key to understanding both the fundamental limitations of standard GNNs and the design principles of more expressive architectures.

1-WL (Colour Refinement). The 1-WL algorithm assigns colours (labels) to nodes and iteratively refines them based on the multiset of neighbour colours:

$$c_v^{(t+1)} = HASH(c_v^{(t)}, \{\{c_u^{(t)} : u \in \mathcal{N}(v)\}\})$$

where HASH is an injective function on pairs (colour, multiset of colours). After stabilisation (at most n steps for n-node graphs), the histogram of final colours forms a graph signature. Two graphs are declared non-isomorphic if their signatures differ. The 1-WL test is: return 'possibly isomorphic' if the signatures match, 'non-isomorphic' if they differ.

Theorem 4.1 (GNN Expressiveness Equals 1-WL, Xu et al., 2019)

Any MPNN with a fixed number of layers L is at most as powerful as the 1-WL test in distinguishing non-isomorphic graphs. Moreover, there exists an MPNN (specifically, GIN — Graph Isomorphism Network) that is as powerful as 1-WL:

$h_v^{(t)} = MLP^{(t)}((1+\varepsilon^{(t)}) h_v^{(t-1)} + \Sigma_{u \in \mathcal{N}(v)} h_u^{(t-1)})$

where $\varepsilon^{(t)}$ is a learnable scalar. GIN is as powerful as 1-WL because its aggregation function is injective on multisets of node features: if the node features of two 1-hop neighbourhoods are different

as multisets, GIN produces different output features. Sum aggregation is injective on multisets of non-negative integer-valued features (and approximately injective for real-valued features); mean and max aggregation are not injective and strictly weaker than 1-WL.

Corollary: standard MPNNs (GCN, GraphSAGE, GAT with mean aggregation) are strictly less powerful than 1-WL. They cannot distinguish: (a) regular graphs of the same degree but different structure; (b) cycles and paths of the same length; (c) certain pairs of non-isomorphic graphs that 1-WL can distinguish.

Limitations of 1-WL. The 1-WL test itself fails to distinguish many pairs of non-isomorphic graphs. The classic example is two regular graphs with the same degree sequence but different structure (e.g., the 3-regular complete bipartite graph K_{3,3} and the 3-regular complete graph K_4, both with 6 nodes and 9 edges; or two 4-regular graphs on 6 nodes). More subtly, 1-WL cannot count cycles (it cannot distinguish graphs that differ only in their cycle structure), cannot count triangles (fundamental for social network analysis), and cannot compute graph properties such as the Weisfeiler-Leman dimension or tree-width.

4.3 The k-WL Hierarchy and Higher-Order GNNs

The k-dimensional Weisfeiler-Leman test (k-WL) operates on tuples of k nodes rather than individual nodes, providing a hierarchy of increasingly powerful graph isomorphism tests. The hierarchy is strict: for each k, there exist graphs that k-WL cannot distinguish but (k+1)-WL can.

k-WL definition. Assign initial colours $c^{(0)}((v_1,...,v_k)) = \text{HASH}(G[\{v_1,...,v_k\}])$ based on the isomorphism type of the induced subgraph on the k-tuple. Then iterate:

$$c^{(t+1)}(v_1,...,v_k) = HASH(c^{(t)}(v_1,...,v_k), \{\{c^{(t)}(v_1,...,v_{i-1},u,v_{i+1},...,v_k) : u \in V\}\} \text{ for } i=1,...,k)$$

Each step aggregates the colours of all k-tuples reachable by replacing one element of the current tuple with any node in the graph. The resulting colour histogram distinguishes more and more graph pairs as k increases.

Theorem 4.2 (WL Hierarchy is Strict)

For each $k \geq 1$, there exist non-isomorphic graphs G and H such that k-WL fails to distinguish them but (k+1)-WL succeeds. Moreover, k-WL is equivalent to the k-variable fragment of first-order logic with counting quantifiers (C_k) for graph properties. The entire hierarchy collapses in terms of decidability at $k = 2$ for vertex-transitive graphs (2-WL is not complete even for strongly regular graphs), and the hierarchy is not known to collapse at any finite k for general graphs.

The practical consequence for GNNs: standard MPNNs are at most 1-WL. Higher-order GNNs operating on k-tuples of nodes are equivalent to k-WL but have computational cost $O(n^k)$, making $k \geq 3$ prohibitively expensive for large graphs. The challenge is to design GNNs that exceed 1-WL expressiveness without the $O(n^k)$ cost.

4.4 Beyond 1-WL: Expressive GNN Architectures

Several architectures have been proposed to exceed the expressiveness of 1-WL without incurring the full $O(n^k)$ cost of higher-order GNNs. Each exploits a different structural feature of graphs that 1-WL cannot capture.

Random Node Initialisation (RNI). Assign each node a random feature from a continuous distribution before running MPNN. With probability 1, all nodes receive distinct features, making the MPNN effectively as powerful as an arbitrary isomorphism test (since distinct initial features break all symmetries). RNI achieves universal expressiveness in expectation at no architectural cost, but its randomness makes the output non-deterministic and requires averaging over multiple runs at inference, increasing cost by a factor equal to the number of samples.

Distance-WL and structural features. Augmenting node features with distance information — the shortest-path distance between each pair of nodes, or the distance from a set of landmark nodes — allows MPNNs to distinguish graphs based on their metric structure. Distance-WL (a GNN variant that uses

pairwise distances as edge features) can distinguish all pairs of graphs that 1-WL cannot distinguish using only local structure. The cost is $O(n^2)$ preprocessing for all-pairs shortest paths (Johnson's algorithm or BFS from each node).

NGNN, OSAN, and subgraph GNNs. A powerful family of approaches computes the representation of each node in the context of subgraphs: for each node v, consider the graph G_v obtained by marking v or deleting it, and run a base MPNN on G_v to obtain node-contextual representations. NGNN (Nested GNN) uses this idea with a hierarchy of subgraph MPNN calls. 3-WL-equivalent architectures can be obtained by considering all induced subgraphs on node pairs (DS-GNN, ESAN). The cost is O(n) base MPNN calls per graph, so $O(n^2)$ total for node-pair subgraphs.

Spectral GNNs and polynomial filters. An orthogonal approach uses the graph Laplacian $L = D - A$ (where D is the degree matrix) to define convolutional filters in the spectral domain. The graph Fourier transform $F = U^T$ (where $L = U\Lambda U^T$) diagonalises convolution, and a graph convolution is a polynomial filter $h(\Lambda) = \Sigma_k \theta_k \Lambda_k$ applied in the spectral domain: $y = U\, h(\Lambda)\, U^T x$. Polynomial filters of degree K are equivalent to K-hop message passing in the spatial domain (due to the Cayley-Hamilton theorem applied to the Laplacian), and spectral GNNs with learnable polynomial filters (ChebNet, GPRGNN, BernNet) are at most 1-WL equivalent. Filters using the full eigenvector basis (SIGN, filtering across all eigenvalues) are more expressive but require the $O(n^3)$ eigendecomposition.

Architecture	WL Power	Node Cost	Edge Cost	Key Mechanism
GCN, GraphSAGE (mean)	< 1-WL	$O(\|E\|)$	$O(\|E\|)$	Mean aggregation (non-injective)
GIN	= 1-WL	$O(\|E\|)$	$O(\|E\|)$	Sum aggregation (injective multisets)

GAT	≤ 1-WL	O(|E|)	O(|E|)	Attention-weighted mean (non-injective)
RNI + GIN	Universal (probabilistic)	O(|E|)	O(|E|)	Random node features
Subgraph GNNs (ESAN)	= 3-WL	O(n|E|)	O(n|E|)	Node-deletion subgraph ensemble
k-GNN (k ≥ 3)	= k-WL	O(n^k)	O(n^k)	k-tuple nodes
Spectral (full eigenbasis)	> 1-WL (partially)	$O(n^3 + n\|E\|)$	$O(n^3)$	Full Laplacian eigenvectors

4.5 Graph Transformers: Attention Meets Topology

The Transformer architecture naturally applies to graph-structured data if we treat the graph as a fully connected set of nodes and use the graph's edges as positional encodings or attention biases. Graph Transformers combine the permutation-equivariance of GNNs with the $O(n^2)$ full-graph attention of Transformers, producing architectures that exceed 1-WL expressiveness while remaining practically trainable for graphs with hundreds to thousands of nodes.

Graphormer. Ying et al. (2021) augment standard multi-head attention with three graph-structural features: (1) a centrality encoding based on degree (in-degree and out-degree for directed graphs) added to node features; (2) a spatial encoding that adds a learnable bias A_{ij} = b_{ϕ(i,j)} to the attention score between nodes i and j, where ϕ(i,j) is the shortest-path distance; (3) an edge encoding that averages learnable edge-type embeddings along the shortest path between i and j. Graphormer strictly surpasses 1-WL (because the distance-based bias encodes metric structure not captured by local neighbourhood aggregation) and achieves state-of-the-art results on molecular property prediction benchmarks.

GPS (General, Powerful, Scalable GNN). Rampášek et al. (2022) propose a principled decomposition of graph learning into two components: a local MPNN for aggregating neighbourhood information (providing strong structural inductive bias) and a global attention layer for long-range interactions (providing expressive power beyond 1-WL). The local and global components are interleaved in each layer, combined with positional encodings (Laplacian eigenvectors, random walk statistics) that encode spectral and topological structure. GPS achieves near-state-of-the-art across all standard graph learning benchmarks while remaining scalable to million-node graphs via sparse attention patterns.

Theorem 4.3 (Graph Transformers are Turing-Universal on Graphs)

A Graph Transformer with full ($O(n^2)$) attention, a sufficient number of layers, and positional encodings that distinguish all nodes (e.g., Laplacian eigenvectors in generic position) can compute any function of the graph that is computable by a Turing machine, up to approximation error. The proof is constructive: the attention mechanism can implement arbitrary set operations on node features; the positional encodings provide unique node identities; and the feed-forward sublayers provide arbitrary pointwise computation. This universality contrasts sharply with standard MPNNs, which are bounded by 1-WL.

The cost of this universality is $O(n^2)$ attention computation. For large graphs ($n > 10{,}000$), this is intractable, and the design of expressive sub-quadratic graph attention is an open research problem. Current approaches include linearised graph attention (replacing softmax attention with kernel methods, as in Section 2.5 of this volume), random sparse attention masks, and hierarchical graph pooling followed by full attention at coarser resolutions.

Conclusions: The Mathematical Logic of Architecture Choice

The four chapters of this volume have developed a coherent framework for understanding neural network architectures as mathematical objects with precise structural properties, rather than engineering constructions chosen by empirical trial. Looking across the territory covered, three master themes emerge.

The first theme is the group-theoretic formulation of inductive biases. Every architecture makes claims about the symmetries of the problem — translation equivariance (CNNs), permutation equivariance (GNNs), rotation equivariance (SE(3)-GNNs), relative position equivariance (RoPE). These claims are either correct (in which case building them in reduces sample complexity proportionally to the group size) or incorrect (in which case they introduce structural bias that hurts generalisation on data that does not respect the assumed symmetry). The design question is not 'what inductive bias is convenient?' but 'what symmetry does the data distribution have?'

The second theme is the expressiveness-efficiency tradeoff, most sharply crystallised in the WL hierarchy for graph neural networks. Increasing expressiveness (from 1-WL to k-WL) costs an exponential factor O(n^k) in computation. The practical resolution — hybrid architectures (GPS, Graphormer), structural positional encodings, subgraph ensembles — achieves super-WL expressiveness at subexponential cost, but always with an overhead relative to the 1-WL baseline. The same tradeoff appears in the sequence modelling context: full attention ($O(L^2)$) versus linear attention ($O(L)$), with selective SSMs (Mamba) and recurrent architectures (RWKV) occupying different points on the expressiveness-efficiency frontier.

The third theme is the memory-computation duality that appears throughout: KV-Cache trades memory for inference speed; Flash Attention trades recomputation for memory; Mamba's parallel scan trades memory-bandwidth for FLOPs; the WL hierarchy trades node-tuple memory for graph

discriminating power. Every efficiency innovation in this chapter is, at its mathematical core, a rearrangement of the memory-computation tradeoff to match the constraints of the available hardware. This is a deep point: the dominant architectures of any generation are determined not only by the data structure of the problems but by the memory hierarchy of the hardware on which they run.

Part III will shift from the structural (what functions can architectures represent?) to the statistical (how do they generalise from finite data?), developing the mathematical theory of statistical learning, including VC dimension, Rademacher complexity, the double descent phenomenon, the Neural Tangent Kernel, Bayesian inference, and uncertainty quantification.

PART III

STATISTICAL LEARNING AND BAYESIAN INFERENCE

A rigorous treatment of learning theory, probabilistic inference, and uncertainty quantification

Preface: The Probabilistic Foundations of Learning

This monograph constitutes the third part of a eight-part treatise on the mathematics of artificial intelligence. It is designed to be read independently of the other volumes, though readers familiar with Parts I and II will recognise recurring themes — particularly the geometry of information and the role of symmetry — that now manifest in probabilistic rather than algebraic clothing.

The central questions addressed here are deceptively simple to state and remarkably deep to answer. When does a learning algorithm trained on finite data reliably predict unseen outcomes? How does one quantify the uncertainty that inevitably accompanies such predictions? When exact inference is intractable — as it almost always is for interesting models — what principled approximation schemes exist, and what guarantees do they carry?

The answer to the first question resides in **statistical learning theory**, which supplies a rigorous vocabulary for generalisation: VC dimension, Rademacher complexity, algorithmic stability, and the tantalising double-descent phenomenon that challenges classical intuitions about model complexity. The answers to the second and third questions reside in the complementary traditions of **Markov chain Monte Carlo** and **variational inference**, two great engines of modern probabilistic computing whose mathematical foundations are surprisingly distinct — one rooted in ergodic theory and stochastic dynamics, the other in convex optimisation and information geometry.

A coherent treatment of all these topics requires the reader to hold simultaneously in mind ideas from measure theory, differential geometry, functional analysis, and stochastic calculus. We attempt to provide the conceptual scaffolding that makes this synthesis possible, keeping rigour always in view while never letting formalism obscure the underlying ideas.

Chapter 1: Statistical Learning Theory

Statistical learning theory asks: under what conditions, and at what sample complexity, can a learning algorithm trained on a finite dataset reliably generalise to new data drawn from the same distribution? The framework we develop — rooted in Valiant's PAC learning model and Vapnik's structural risk minimisation — provides answers whose form is universal, applying to any hypothesis class and any loss function satisfying mild regularity conditions.

1.1 The PAC Learning Framework

Let $\mathcal{X}$ be an input space, $\mathcal{Y}$ an output space, and $\mathcal{H}$ a hypothesis class of functions $h: \mathcal{X} \to \mathcal{Y}$. A concept $c: \mathcal{X} \to \mathcal{Y}$ is to be learned from a sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ drawn i.i.d. from an unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. The **true risk** of a hypothesis is $L(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(h(x), y)]$ and the **empirical risk** is $\hat{L}(h) = (1/m)\,\Sigma_i\, \ell(h(x_i), y_i)$.

A learning algorithm A **PAC-learns** $\mathcal{H}$ if for every $\varepsilon, \delta > 0$ and every distribution $\mathcal{D}$, given a sample of size $m \geq m(\varepsilon, \delta)$, with probability at least $1 - \delta$ over the draw of S, the output hypothesis satisfies $L(A(S)) \leq \min_{h\in\mathcal{H}} L(h) + \varepsilon$. The quantity $m(\varepsilon, \delta)$ is the **sample complexity** — the number of examples needed to achieve accuracy ε with confidence $1 - \delta$.

Theorem 1.1 — Finite Hypothesis Class Bound

Let $\mathcal{H}$ be finite, $\ell \in [0,1]$. For any $\delta > 0$, with probability $\geq 1-\delta$ over $S \sim \mathcal{D}^m$, for all $h \in \mathcal{H}$:

$$L(h) \leq \hat{L}(h) + \sqrt{\left(\ln|\mathcal{H}| + \ln(1/\delta)\right) / (2m)}$$

Proof: By a union bound over $|\mathcal{H}|$ Hoeffding inequalities, one per hypothesis. The $|\mathcal{H}|$-term measures the complexity penalty for searching through the class. The bound is tight for the ERM on uniformly bad instances.

For infinite hypothesis classes, the finite-class bound is inapplicable. The first key insight of VC theory is that the effective size of a hypothesis class for a binary classification problem is governed not by its cardinality but by its capacity to shatter finite sets of points — the VC dimension.

1.2 VC Dimension and the Fundamental Theorem of Learning

The **VC dimension** $d_{VC}(\mathcal{H})$ is the largest integer d such that there exists a set of d points that $\mathcal{H}$ can shatter — label in all 2^d possible ways. A hypothesis class with

$d_{VC}(\mathcal{H}) = \infty$ is not PAC-learnable (in the distribution-free sense). The Sauer-Shelah lemma controls the growth of the shatter function:

Lemma 1.2 — Sauer–Shelah

If d_VC($\mathcal{H}$) = d, then the growth function satisfies:

$$\Pi_{\mathcal{H}}(m) = \max_{S \subseteq \mathcal{X},\, |S|=m} |\{h|_S : h \in \mathcal{H}\}| \le \Sigma_{i=0}^{d} C(m,i) \le (em/d)^d$$

The key transition: for $m \le d$, $\Pi_{\mathcal{H}}(m) = 2^m$ (exponential); for $m > d$, $\Pi_{\mathcal{H}}(m)$ grows polynomially. This phase transition controls learnability.

Theorem 1.3 — Fundamental Theorem of PAC Learning (VC)

For a binary classification class $\mathcal{H}$ with $d = d_{VC}(\mathcal{H}) < \infty$, ERM PAC-learns $\mathcal{H}$ with sample complexity:

$$m(\varepsilon, \delta) = \Theta\left((d + \ln(1/\delta)) / \varepsilon^2 \right)$$

Upper bound proof: Symmetrisation (ghost sample), then union over $\Pi_{\mathcal{H}}(2m)$ dichotomies, then Hoeffding. Lower bound: Sauer-Shelah + a specific hard distribution on a shattered set.

The generalisation bound implied by the VC theorem predicts that test error decreases as model complexity increases until a sweet spot, then rises. This underpins the classical bias-variance tradeoff. But empirical observation of modern over-parameterised networks violates this prediction in a systematic way — a discrepancy resolved by the double-descent phenomenon.

1.3 Rademacher Complexity

Rademacher complexity provides a data-dependent generalisation bound, often tighter than VC bounds. Define the **empirical Rademacher complexity**:

$$\hat{R}_S(\in) = {}_{i}\,_\sigma[\, sup_\{h_i\}\, (1/m)\, \Sigma\;\; {}_{i}\, \sigma \not{h}(x \not{)}]$$

where $\sigma_i \sim Uniform(\{\pm 1\})$ are independent Rademacher variables. Intuitively, $\hat{R}_S(\mathcal{H})$ measures how well the best hypothesis in $\mathcal{H}$ can correlate with random noise — a higher value means more capacity to fit arbitrary labels.

Theorem 1.4 — Rademacher Generalisation Bound

For any $\mathcal{H}$ of functions $\mathcal{H} \to [0,1]$ and any $\delta > 0$, with probability $\ge 1-\delta$ over $S \sim \mathcal{D}^m$, for all $h \in$

$\mathscr{H}$:

$$L(h) \leq \hat{L}(h) + 2R_m(\mathscr{H}) + \sqrt{(\ln(1/\delta) / (2m))}$$

where $R_m(\mathscr{H}) = \mathbb{E}_S[\hat{R}_S(\mathscr{H})]$ is the expected Rademacher complexity. Unlike VC bounds, this captures the interaction between $\mathscr{H}$ and the data distribution $\mathscr{D}$.

For neural networks, Rademacher bounds can be expressed in terms of operator norms of weight matrices. For a depth-L network with weight matrices $W_1, \ldots, W_L$ and 1-Lipschitz activations:

$$R_m(\not{net}) \leq (2^{3/2} B_x) / \sqrt{m} \cdot \Pi_{l=1}^{L} \|W_l\|_{op} \cdot \sqrt{(\Sigma_l \|W_l\|_F^2 / \|W_l\|_{op}^2)}$$

where $B_x = max\ \|x\|$. This bound degrades with depth and the product of spectral norms, explaining why spectral normalisation (constraining $\|W_l\|_{op} \leq 1$) is an effective regularisation technique. However, modern large networks routinely violate the conditions under which these bounds are non-vacuous, which motivates the algorithmic stability and PAC-Bayes approaches.

1.4 Algorithmic Stability and PAC-Bayes

Bousquet and Elisseeff's **algorithmic stability** framework measures generalisation through the sensitivity of an algorithm's output to replacing a single training point. Algorithm A is β-uniformly stable if for all datasets S, S' differing in one example:

$$sup_{(x,y)} |\ell(A(S), x, y) - \ell(A(S'), x, y)| \leq \beta$$

Theorem 1.5 — Stability Generalisation Bound

If A is β-uniformly stable with $\beta \leq 1/(2m)$, then for any $\delta > 0$, with prob. $\geq 1-\delta$:

$$L(A(S)) \leq \hat{L}(A(S)) + 2\beta + (4m\beta + 1)\sqrt{(\ln(1/\delta) / (2m))}$$

SGD with step size η satisfies $\beta = O(\eta T/m)$ after T steps. This connects optimisation trajectory length to generalisation, explaining why early stopping helps.

PAC-Bayes bounds take a Bayesian perspective. Given a prior P over hypotheses and a posterior Q chosen after seeing data:

Theorem 1.6 — McAllester PAC-Bayes Bound

For any prior P, with probability ≥ 1−δ over $S \sim \mathcal{D}^m$, for all posteriors Q:

$$\mathbb{E}_Q[L(h)] \leq \mathbb{E}_Q[\hat{L}(h)] + \sqrt{\left(D_{KL}(Q \| P) + \ln(m/\delta)\right) / (2m-1)}$$

This bound is non-vacuous even for infinite hypothesis classes because the KL term controls the effective complexity. PAC-Bayes is the only framework giving tight (single-digit percentage) error bounds for deep networks.

1.5 Double Descent and Benign Overfitting

The classical bias-variance tradeoff predicts a U-shaped test error curve as a function of model complexity: bias decreases while variance increases, and there exists an optimal complexity at the interpolation point. The double-descent phenomenon, empirically documented across linear models, kernel machines, and neural networks, reveals a **second descent** of test error beyond the interpolation threshold, fundamentally challenging classical learning theory.

The interpolation threshold corresponds to the number of parameters required to exactly fit the training data (training error = 0). Classical theory predicts diverging test error at this point. Instead, empirically: for interpolating models beyond this threshold, test error decreases again, approaching or surpassing the classical optimum.

Theorem 1.7 — Benign Overfitting in Linear Models (Bartlett et al. 2020)

Consider the minimum-ℓ^2-norm interpolating estimator $\hat{\beta} = X^\dagger y$ for $X \in \mathcal{D}^{n \times p}$ with $p > n$.

Under Gaussian design with effective rank $r_{eff} = (\sum_i \lambda_i)^2 / \sum_i \lambda_i^2$ of the covariance Σ:

$$\|\hat{\beta} - \beta^*\|_\Sigma^2 = O(n / r_{eff} + \text{noise} \cdot p/n)$$

Generalisation is benign (risk → 0 as $n,p \to \infty$) if and only if $r_{eff}/n \to \infty$ AND $p/n \to \infty$.

The condition $r_{eff}/n \to \infty$ requires the spectrum of Σ to be spread — a "spiked" spectrum leads to malignant overfitting.

The mechanism behind benign overfitting is the **implicit regularisation** of gradient descent: with zero initialisation, gradient descent converges to the minimum-ℓ^2-norm interpolating solution in the underdetermined linear case. For neural networks, the analogous result holds in the **Neural Tangent Kernel** regime: gradient flow converges to the minimum-NTK-norm interpolating function. The geometry of the NTK — its

effective rank and eigenspectrum — determines whether overfitting is benign or malignant.

1.6 The Neural Tangent Kernel

At infinite width, a randomly initialised neural network $f_\theta: \mathbb{R}^{n_0} \to \mathbb{R}$ converges to a Gaussian process, and during training with gradient flow the parameters barely move from initialisation — the **lazy training** or **NTK regime**. The Neural Tangent Kernel (Jacot et al. 2018) is:

$$\Theta(x, x') = \Sigma_p \, (\partial f_\theta(x)/\partial \theta_p)(\partial f_\theta(x')/\partial \theta_p)$$

Theorem 1.8 — NTK Convergence

For a fully-connected network of width n with Lipschitz activations σ, as n → ∞:

(i) The NTK Θ(θ, x, x') converges in probability to a deterministic kernel Θ∞(x, x')

(ii) Under gradient flow, Θ(θ(t), ·, ·) = Θ∞(·, ·) + O(1/√n) for all t ≥ 0

(iii) The network function f_θ(t) evolves as: df/dt = −Θ∞ (f − y), giving f(t) = (I−e^{−Θ∞t}) (Θ∞)⁻¹y

Convergence rate: min eigenvalue of Θ∞ on training set determines exponential convergence rate of training loss.

The NTK regime is mathematically clean but practically limited. In finite-width networks, features are learned — the kernel changes during training — and this **feature learning** is responsible for the empirical superiority of trained networks over kernel machines with fixed NTK. The catapult phase (initial rapid kernel change before lazy settling) and the edge-of-stability phenomenon (loss oscillating near the sharpness threshold $2/\eta$ with η the learning rate) are active research frontiers beyond the NTK theory.

Chapter 2: Optimisation and Loss Landscape

The loss landscape of a neural network is a high-dimensional function whose global geometry determines which optimisation algorithms succeed. The landscape of modern over-parameterised networks is profoundly non-convex, yet in practice gradient-based methods reliably find solutions with near-zero training loss that also generalise. Understanding why requires a synthesis of dynamical systems theory, random matrix theory, and topological analysis.

2.1 Stochastic Gradient Descent as a Dynamical System

Let $L(\theta) = \mathbb{E}_{z\sim\mathcal{D}}[\ell(f_\theta, z)]$ be the population loss. Gradient descent iterates $\theta_{t+1} = \theta_t - \eta\nabla L(\theta_t)$ but computing ∇L exactly is infeasible. SGD uses a mini-batch estimate:

$$\theta_{t+1} = \theta_t - \eta\,\hat{g}_t, \quad \text{where } \hat{g}_t = \frac{1}{|B_t|}\sum_{z\in B_t}\nabla\ell(f_{\theta_t}, z)$$

The noise $\hat{g}_t - \nabla L(\theta_t)$ has zero mean (unbiased) and covariance approximately $C(\theta_t)/|B|$ where $C(\theta) = \mathbb{E}[\nabla\ell\,\nabla\ell^T] - \nabla L\,\nabla L^T$ is the gradient covariance. In continuous time, SGD is approximated by the Langevin SDE:

$$d\theta = -\nabla L(\theta)\,dt + \sqrt{(\eta/|B|)}\,C(\theta)^{1/2}\,dW_t$$

This connection to Langevin dynamics reveals that SGD implicitly samples from the Gibbs distribution $\pi(\theta) \propto \exp(-L(\theta)/T)$ with temperature $T = \eta/(2|B|)$. Larger learning rates or smaller batches correspond to higher temperature — more exploration. The flat-minimum preference of SGD (its tendency to converge to wide valleys in the loss landscape rather than sharp minima) arises from this temperature: sharp minima occupy small measure under the Gibbs distribution at any positive temperature.

Theorem 2.1 — SGD Flat-Minimum Bias (Zhu et al. 2019)

Let θ^* be a local minimum. The expected escape time from a δ-neighbourhood scales as:

$$\tau_{\text{escape}} \propto \exp(\Delta L / T) = \exp(|B|\,\Delta L / \eta)$$

where ΔL is the loss barrier height at the boundary. Flatter minima (same depth but wider basin) have lower barriers in most directions, hence shorter escape time — SGD spends more time in flat basins proportional to their volume × depth tradeoff.

2.2 Adaptive Methods: Adam, AdaGrad, and RMSProp

The pathologies of SGD — sensitivity to learning rate, slow convergence in directions of small curvature — motivated adaptive methods that rescale each gradient coordinate by a running estimate of its magnitude. The unifying framework:

$$\theta_{t+1} = \theta_t - \eta \times m_t \oslash (v_t^{1/2} + \varepsilon)$$

where m_t is the first moment estimate (gradient), v_t is the second moment estimate (squared gradient), and $\oslash$ is element-wise division. **AdaGrad** sets $v_t = \Sigma_{s\le t}\, \hat{g}_s^2$ (cumulative sum — decreasing effective LR, never recovers); **RMSProp** uses an exponential moving average $v_t = \beta v_{t-1} + (1-\beta)\hat{g}_t^2$; **Adam** combines both:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\hat{g}_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2)\hat{g}_t^2$$

$$\hat{m}_t = m_t/(1-\beta_1^t), \quad \hat{v}_t = v_t/(1-\beta_2^t) \quad \textit{(bias correction)}$$

AdamW separates weight decay from the adaptive scaling: instead of adding $\lambda\theta$ to the gradient before adaptation (which shrinks the effective weight decay for high-variance parameters), it applies $\theta_t \leftarrow (1 - \lambda\eta)\theta_{t-1}$ as a multiplicative step. This decoupling is crucial for correct regularisation of embedding layers and attention weights, which typically have much smaller gradient variance than MLP weights.

Theorem 2.2 — Adam Convergence (Non-convex)

For L smooth ($\|\nabla^2 L\| \le \beta$) with gradients g_t having bounded variance σ^2:

$$(1/T)\ \Sigma_{t=1}^{T}\ \mathbb{E}[\|\nabla L(\theta_t)\|^2] \le O(\sqrt{(d \ln T)} / \sqrt{T})$$

where d is the parameter dimension. The ln T factor (logarithmic regret) arises from the time-varying denominator $\hat{v}_t^{1/2}$.

Caveat: This bound is for fixed β_1, β_2 and does not account for the warm-up phase. Empirically, the first few thousand steps are critical and not well-modelled by asymptotic theory.

2.3 Loss Landscape Geometry: Saddle Points, Flat Minima, and the Mode Connectivity

In high-dimensional non-convex optimisation, the dominant obstruction to convergence is not local minima but **saddle points** — critical points where the Hessian has both positive and negative eigenvalues. In a p-dimensional landscape, a random critical point is a local minimum with probability exponentially small in p. Almost all critical points with loss $L > L^*$ are saddle points.

Theorem 2.3 — Saddle Point Avoidance (Lee et al. 2016)

For gradient flow on a Morse function (all critical points non-degenerate), the set of initial conditions converging to a strict saddle point has Lebesgue measure zero. More strongly, for smooth gradient descent with appropriate step size, all stable limit points of gradient descent are local minima.

Practical implication: SGD noise (or any perturbation) guarantees escape from saddle points, justifying the empirical observation that SGD finds minima rather than getting stuck at saddles.

Mode connectivity (Draxler et al. 2018, Garipov et al. 2018) reveals that independently trained solutions in the loss landscape are connected by paths of approximately constant loss. Specifically, for overparameterised networks there exist piecewise-linear paths $\gamma: [0,1] \to \mathbb{R}^p$ between any two converged solutions θ_A and θ_B such that $L(\gamma(t)) \approx \max(L(\theta_A), L(\theta_B))$ for all t. This mode connectivity implies that the loss landscape does not fragment into isolated good solutions but rather contains a large connected manifold of good minima — the **solution manifold hypothesis**.

The sharpness-awareness of a minimum is measured by the maximum eigenvalue of the Hessian $\lambda_{max}(\nabla^2 L(\theta))$. SAM (Sharpness-Aware Minimisation) solves:

$$\min_\theta \max_{\|\varepsilon\| \le \rho} L(\theta+\varepsilon) \quad \text{approximately:} \quad \min_\theta L(\theta) + \rho\|\nabla L(\theta)\|$$

The perturbation $\hat{\varepsilon} = \rho \nabla L / \|\nabla L\|$ maximising local loss has a clean geometric interpretation: it pushes θ along the steepest ascent direction on the loss surface, forcing the optimiser to find flatlands rather than ridges. SAM's efficacy is partly explained by a PAC-Bayes bound: the expected loss under a spherical Gaussian perturbation of radius ρ is a valid generalisation bound, and SAM directly minimises this quantity.

2.4 The Edge of Stability

An unexpected phenomenon in full-batch gradient descent — the **edge of stability** (Cohen et al. 2021) — challenges the standard convergence theory. For learning rate η, gradient descent is classically guaranteed to decrease loss when $\lambda_{max}(\nabla^2 L) < 2/\eta$. Yet experiments show that during training, λ_{max} rises precisely to $2/\eta$ and hovers there, while the loss still decreases. The mechanism:

(1) **Progressive sharpening**: gradient descent naturally moves towards sharper regions of the landscape; (2) At the edge $\lambda_{max} \approx 2/\eta$, the dynamics enter a regime of **loss oscillations** with period approximately 2 (chaotic but loss-decreasing on

average); (3) The bulk structure of the Hessian (not just λ_max) controls the effective step taken in each direction, allowing stable descent in low-curvature subspaces even when the sharpest direction is unstable.

Chapter 3: Kernel Methods — RKHS, SVM, and Maximum Mean Discrepancy

Kernel methods provide a principled framework for non-linear learning through a central algebraic device: representing the model as a function in a Reproducing Kernel Hilbert Space defined implicitly by a positive-definite kernel function. This framework — now understood as a classical limit of infinitely wide neural networks — achieves optimal generalisation bounds with explicit geometric interpretation.

3.1 Reproducing Kernel Hilbert Spaces

A **reproducing kernel Hilbert space** (RKHS) $\mathscr{H}_k$ associated with a positive-definite kernel $k: \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ is a Hilbert space of functions $f: \mathscr{X} \to \mathbb{R}$ satisfying the reproducing property:

$$f(x) = \langle f, k(\cdot, x) \rangle_{\mathscr{H}_k} \quad \text{for all } f \in \mathscr{H}_k \text{ and } x \in \mathscr{X}$$

This property has profound consequences. First, **Mercer's theorem** provides a spectral representation: for a continuous positive-definite kernel on a compact domain, there exist eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots > 0$ and eigenfunctions ψ_j such that $k(x, x') = \sum_j \lambda_j \psi_j(x) \psi_j(x')$. The RKHS is then the image of ℓ^2 under the map $(c_j) \mapsto \sum_j c_j \sqrt{\lambda_j}\, \psi_j$. Second, the **kernel trick** allows computing inner products in $\mathscr{H}_k$ without explicit feature maps:

$$\langle \varphi(x), \varphi(x') \rangle_{\mathscr{H}} = k(x, x') \quad \text{where} \quad \varphi(x) = k(\cdot, x) \in \mathscr{H}_k$$

Theorem 3.1 — Representer Theorem (Schölkopf–Smola–Müller)

For any loss $L(f(x_1), \ldots, f(x_m), y_1, \ldots, y_m)$ and RKHS regulariser $\Omega(\|f\|_{\mathscr{H}_k})$:

$$f^* = \operatorname{argmin}_{f \in \mathscr{H}_k} L(f(x_1), \ldots, f(x_m), y) + \Omega(\|f\|_{\mathscr{H}_k})$$

has the finite expansion: $f^*(\cdot) = \sum_{i} \alpha_i\, k(\cdot, x_i)$

Proof: Any component of f orthogonal to $\operatorname{span}\{k(\cdot, x_i)\}$ neither changes the loss (which depends only on evaluations at training points) nor decreases $\|f\|$, so the optimum has zero orthogonal component.

Common kernels and their RKHS interpretation: the **Gaussian RBF** $k(x,x') = \exp(-\|x-x'\|^2/2\sigma^2)$ corresponds to an infinite-dimensional feature space (entire function class of analytic functions); the **polynomial kernel** $k(x,x') = (x^T x' + c)^d$ corresponds to

all monomials of degree up to d; the **arc-cosine kernel** $k_n(x,x')$ of Cho and Saul is exactly the NTK of an infinite-width network with specific activations.

3.2 Support Vector Machines and Structural Risk Minimisation

The Support Vector Machine arises from Vapnik's **structural risk minimisation** principle: instead of minimising empirical risk alone (ERM), jointly minimise empirical risk and a measure of model complexity. For binary classification with labels $y \in \{\pm 1\}$, the hard-margin SVM seeks:

$$\min_{w,\,b} \ \tfrac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i(w \cdot x_i + b) \geq 1 \ \ \forall i$$

The geometric interpretation: the constraint ensures that all training points lie outside a margin band of width $2/\|w\|$ around the decision boundary. Minimising $\tfrac{1}{2}\|w\|^2$ therefore **maximises the margin**. The dual problem:

$$\max_{\alpha} \ \sum_i \alpha_i - \tfrac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \quad \text{s.t.} \ \ \alpha_i \geq 0, \ \ \sum_i \alpha_i y_i = 0$$

The KKT conditions show that only points on the margin boundary (support vectors) have $\alpha_i > 0$ — the model complexity depends only on the number of support vectors, not the total training set size. The soft-margin extension (C-SVM) introduces slack variables $\xi_i \geq 0$ and penalty $C \sum_i \xi_i$, trading margin width for tolerance of misclassification.

Theorem 3.2 — SVM Generalisation Bound via Margin

For a linear classifier with margin ϱ, with probability $\geq 1-\delta$:

$$L(h) \leq \hat{L}_\gamma(h) + O\left(\hat{R}(\|w\|/\gamma) + \sqrt{(\ln(1/\delta)/m)}\right)$$

where $\hat{L}_\gamma$ is the γ-margin training error and $\hat{R}$ is the Rademacher complexity bounded by the margin. This shows SVMs directly minimise the generalisation-relevant quantity, not just empirical risk.

3.3 Maximum Mean Discrepancy

Maximum Mean Discrepancy (MMD) is a kernel-based measure of distributional distance with applications from two-sample testing to generative modelling evaluation. Given probability distributions P and Q on $\mathcal{X}$, and a unit ball in RKHS $\mathcal{H}_k$:

$$\mathrm{MMD}^2(P, Q) = \sup_{f \in \mathcal{H}_k,\ \|f\| \leq 1} \left(\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{y \sim Q}[f(y)]\right)^2$$

The variational definition yields a computable estimator: $MMD^2(P,Q) = \mathbb{E}_{x,x' \sim P}[k(x,x')] - 2\mathbb{E}_{x \sim P, y \sim Q}[k(x,y)] + \mathbb{E}_{y,y' \sim Q}[k(y,y')]$. This is a U-statistic with well-understood asymptotic distribution under $H_0: P = Q$ (the MMD statistic follows a weighted sum of χ^2_1 distributions under the null), enabling rigorous hypothesis testing.

Theorem 3.3 — MMD Metrises Weak Convergence

For a characteristic kernel k (e.g., Gaussian RBF), $\mathrm{MMD}(P,Q) = 0 \iff P = Q$.

Furthermore, $\mathrm{MMD}(P_n, P) \to 0$ iff $P_n \Rightarrow P$ (weak convergence) when $\mathcal{X}$ is compact.

This distinguishes MMD from moment-matching: characteristic kernels capture ALL moments of the distribution simultaneously, not just finite-order ones.

In machine learning, MMD appears as: (1) the training objective for Maximum Mean Discrepancy Networks (a precursor to GANs); (2) the divergence in InfoVAE, replacing the KL term in the ELBO; (3) a diagnostic for covariate shift and domain adaptation; (4) the Stein discrepancy (a related kernel measure) appears in SVGD (Stein Variational Gradient Descent), a particle-based approximate inference method.

3.4 Kernel Stein Discrepancy and SVGD

The **Kernelised Stein Discrepancy** (KSD) measures how far a distribution q is from a target p without requiring samples from p — only access to $\nabla \log p$. For Stein operator $\nabla_p f(x) = \nabla_x \log p(x)\, f(x) + \nabla_x f(x)$:

$$KSD^2(q, p) = \mathbb{E}_{x,x' \sim q}[u_p(x, x')]$$

where the kernel Stein function $u_p(x, x')$ depends only on $\nabla \log p$ and k. This enables testing whether $q = p$ using only samples from q — powerful for evaluating MCMC convergence without knowing p's normalisation constant.

Stein Variational Gradient Descent (SVGD, Liu and Wang 2016) uses functional gradient descent on the KL divergence: starting from particles $x_i \sim q_0$, the update $x_i \leftarrow x_i + \varepsilon\, \varphi^*(x_i)$ with optimal velocity field $\varphi^*(x) = \mathbb{E}_{y \sim q}[k(y,x) \nabla_y \log p(y) + \nabla_y k(y,x)]$ yields the steepest descent in W_2 distance on the manifold of smooth densities. SVGD bridges particle methods and variational inference: with one particle it becomes MAP estimation; with many particles it converges to the posterior.

Chapter 4: Markov Chain Monte Carlo

Monte Carlo methods address the fundamental challenge of Bayesian computation: given a posterior distribution $\pi(\theta) \propto p(\theta) \times p(\text{data}|\theta)$ that is analytically intractable, how do we compute expectations $\mathbb{E}_\pi[f(\theta)]$? Markov chain Monte Carlo constructs an ergodic Markov chain with stationary distribution π, exploiting the ergodic theorem to convert time averages into space averages. The various MCMC algorithms differ in how they construct this chain, with fundamental tradeoffs between mixing time, computational cost per step, and applicability.

4.1 Metropolis-Hastings: The Master Algorithm

The Metropolis-Hastings (MH) algorithm constructs a reversible Markov chain on any measurable space by accepting or rejecting proposed moves with a carefully designed acceptance probability. Given proposal kernel $q(\theta' | \theta)$, from current state θ_t:

$\theta' \sim q(\cdot | \theta_t)$ (propose)

$\alpha = \min(1, \pi(\theta')q(\theta_t|\theta')/\pi(\theta)q(\theta'|\theta_t))$ (acceptance ratio)

$\theta_{t+1} = \theta'$ with prob α, else θ_t (accept/reject)

Theorem 4.1 — MH Correctness

The MH chain satisfies detailed balance: $\pi(\theta)T(\theta'|\theta) = \pi(\theta')T(\theta|\theta')$ for all θ, θ'.

Proof: For $\alpha < 1$ states, $\pi(\theta)q(\theta'|\theta)\alpha(\theta,\theta') = \pi(\theta')q(\theta|\theta') \times \pi(\theta)q(\theta'|\theta)/\pi(\theta')q(\theta|\theta') = \pi(\theta)q(\theta'|\theta)$, and $\alpha(\theta',\theta) = 1$ by symmetry of the ratio, giving detailed balance. The chain is π-irreducible if $\text{supp}(q(\cdot|\theta)) = \text{supp}(\pi)$ for all θ.

The critical design choices are the proposal q and the scale at which it operates. For a Gaussian random walk proposal with covariance $\Sigma\Delta$, the optimal acceptance rate is approximately 23.4% in high dimensions (Roberts-Gelman-Gilks 1997), achieved with $\Sigma\Delta = (2.38^2/d)\,\Sigma_\pi$ where Σ_π is the posterior covariance. This scaling law reveals a fundamental tension: proposals that are too narrow mix slowly (high acceptance rate but small moves); proposals that are too wide rarely accept. The RWMH mixing time scales as $O(d)$ for isotropic posteriors — polynomial in dimension, but prohibitive for thousands of parameters.

4.2 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) escapes the random-walk scaling by proposing moves along Hamiltonian trajectories. Augment the parameter space with auxiliary momentum $p \sim \mathcal{N}(0, M)$ and define the Hamiltonian:

$$H(\theta, p) = -\log \pi(\theta) + \tfrac{1}{2} p^T M^{-1} p \quad \text{(potential + kinetic energy)}$$

Hamiltonian dynamics $d\theta/dt = \partial H/\partial p = M^{-1}p$, $dp/dt = -\partial H/\partial \theta = \nabla \log \pi(\theta)$ preserve both H (energy conservation) and volume (Liouville's theorem). Discretising with the leapfrog integrator:

$$p_{t+\varepsilon/2} = p_t + (\varepsilon/2)\nabla \log \pi(\theta_t)$$

$$\theta_{t+\varepsilon} = \theta_t + \varepsilon M^{-1} p_{t+\varepsilon/2}$$

$$p_{t+\varepsilon} = p_{t+\varepsilon/2} + (\varepsilon/2)\nabla \log \pi(\theta_{t+\varepsilon})$$

After L leapfrog steps, the proposed (θ', p') is accepted via an MH step correcting for discretisation error. For well-tuned step size ε, acceptance rates are close to 1 (typically 65-80%), and the mixing time scales as $O(d^{1/4})$ — a dramatic improvement over RWMH's $O(d)$.

Theorem 4.2 — HMC Mixing (Mangoubi-Smith 2019)

For log-concave target π with condition number κ (ratio of max to min curvature), HMC with leapfrog integration mixes in:

$O(\kappa^{1/2} d^{1/4})$ steps (vs $O(\kappa d)$ for Langevin, $O(\kappa d)$ for RWMH)

The condition number κ measures posterior anisotropy; adaptive preconditioning (using estimated $M = \Sigma_\pi$) reduces the effective κ to 1.

4.3 NUTS: No-U-Turn Sampler

The No-U-Turn Sampler (Hoffman-Gelman 2014) automates the critical tuning parameters of HMC — step size ε and trajectory length L. The number of leapfrog steps L is chosen adaptively by detecting when the trajectory begins to "turn around" (the U-turn criterion):

$$\text{stop when } (\theta_L - \theta_0) \cdot p_L < 0 \text{ or } (\theta_L - \theta_0) \cdot p_0 < 0$$

NUTS builds a binary tree of proposals by doubling the trajectory in the direction that does not violate the U-turn criterion, then samples uniformly from the resulting set (the "slice" in the joint (θ, p) space). The resulting algorithm satisfies detailed balance and is provably at least as efficient as optimal fixed-length HMC. Dual averaging

(Nesterov 2009) adapts the step size ε during a warmup phase to achieve a target acceptance probability (typically 0.8).

NUTS is the default sampler in Stan and PyMC, enabling Bayesian inference on complex hierarchical models with thousands of parameters. Its main limitation is the gradient computation cost per step — for large neural networks with millions of parameters, NUTS is computationally prohibitive, motivating the SGLD approach.

4.4 Stochastic Gradient Langevin Dynamics

SGLD (Welling and Teh 2011) brings Bayesian deep learning into computational reach by replacing the exact gradient with a mini-batch estimate and eliminating the MH accept/reject step. Starting from the overdamped Langevin SDE:

$$d\theta = \tfrac{1}{2}\nabla \log \pi(\theta)\, dt + dW_t \implies \theta_{t+1} = \theta_t + (\varepsilon_t/2)\nabla \log \pi(\theta_t) + \sqrt{\varepsilon_t}\, \xi_t$$

where $\xi_t \sim \mathcal{N}(0, I)$. Replacing $\nabla \log \pi$ with a stochastic estimate on a mini-batch B_t:

$$\nabla \log \pi(\theta) = \nabla \log p(\theta) + (N/|B|) \sum_{i \in B} \nabla \log p(x_i \mid \theta)$$

Theorem 4.3 — SGLD Convergence

Under log-concavity and Lipschitz gradient assumptions, SGLD with step size $\varepsilon \propto t^{-1/3}$ satisfies:

$$W_2(\rho_t, \pi) \le O(t^{-1/3} + \text{noise_bias})$$

where noise_bias comes from mini-batch gradient noise. The step size must be annealed to zero for exact convergence; a fixed step size yields a biased stationary distribution within $O(\varepsilon)$ of π in W_2.

In practice: use fixed ε as an approximate Bayesian sampler and diagnose convergence via Stein discrepancy or kernel two-sample tests.

4.5 Parallel Tempering and Replica Exchange

Highly multimodal posteriors — common in neural network parameter spaces, mixture models, and protein folding — defeat all unimodal MCMC algorithms by creating bottlenecks at low-probability barriers between modes. **Parallel tempering** (replica exchange MCMC) simultaneously runs chains at different temperatures $T_1 < T_2 < \ldots < T_K$ targeting the tempered distributions $\pi_k(\theta) \propto \pi(\theta)^{1/T_k}$.

At high temperature, barriers are smoothed and modes mix freely. Periodically, adjacent chains attempt swap moves with acceptance probability:

$$\alpha_{swap} = \min(1, [\pi_{k+1}(\theta_k)/\pi_k(\theta_k)] \times [\pi_k(\theta_{k+1})/\pi_{k+1}(\theta_{k+1})])$$

$$= \min(1, \exp((1/T_{k+1} - 1/T_k)(\log \pi(\theta_k) - \log \pi(\theta_{k+1}))))$$

Accepted swaps propagate low-temperature solutions that were found at high temperature, enabling the cold chain to jump between modes that are separated by barriers too high for direct Metropolis moves. The optimal temperature ladder (for spherical Gaussians) is geometric: $T_k = T_1^{(K-k)/(K-1)}$, which ensures roughly constant swap rates of approximately 23.4%. For real-world posteriors, adaptive temperature scheduling based on measured swap rates is standard.

Theorem 4.4 — Parallel Tempering Mixing Enhancement

For a bimodal target with mode separation Δ and barrier height B, RWMH mixing time is O(exp(B/T)).

Parallel tempering with K temperatures mixing to the cold chain in $O(K \times \text{mixing_at_}T_1)$ steps, where $T_1 \geq B$ means the hottest chain mixes in O(poly(d)) steps.

Optimal K = O(log(B/T_cold) × d^{1/2}) temperature levels achieve polynomial mixing.

Chapter 5: Variational Inference

Variational inference transforms posterior computation from a sampling problem into an optimisation problem. By restricting the posterior approximation to a tractable family and maximising a lower bound on the log evidence, variational methods scale to large datasets and complex models where MCMC is computationally prohibitive. The price paid is approximation bias from the restricted family; the gain is computational speed of several orders of magnitude.

5.1 The Evidence Lower Bound (ELBO)

For a model with observations x, latent variables z, and parameters θ, the log-evidence $\log p_\theta(x)$ is intractable because it requires integrating over z. The ELBO decomposes this quantity:

$$\log p_\theta(x) = \mathbb{E}_{q_\varphi(z|x)}[\log p_\theta(x,z) - \log q_\varphi(z|x)] + D_{KL}(q_\varphi(z|x) \,\|\, p_\theta(z|x))$$

$$\geq \mathcal{L}(\theta, \varphi; x) \equiv \mathbb{E}_{q_\varphi}[\log p_\theta(x|z)] - D_{KL}(q_\varphi(z|x) \,\|\, p(z))$$

The ELBO decomposes as a reconstruction term (expected log-likelihood) minus a KL regulariser. The bound is tight when $q_\varphi(z|x) = p_\theta(z|x)$ — the exact posterior — which is the condition that VI is trying to approximate. Maximising over θ and φ simultaneously performs both model learning and posterior approximation.

The KL divergence $D_{KL}(q_\varphi \,\|\, p)$ in the ELBO uses the **forward KL** (also called exclusive or mean-seeking). Optimising this form encourages q_φ to be zero wherever p is zero, and to cover all modes of p — resulting in broad, diffuse approximations that may miss fine-grained posterior structure. The alternative **reverse KL** $D_{KL}(p \,\|\, q_\varphi)$ (inclusive, mass-covering) penalises q_φ for being small where p is large, producing mode-seeking behaviour. Most VI literature uses forward KL for computational tractability.

5.2 Coordinate Ascent Variational Inference (CAVI)

Under the **mean-field assumption**, the variational posterior factorises over J disjoint groups: $q(z) = \prod_{j=1}^{J} q_j(z_j)$. CAVI optimises each factor in turn while holding others fixed:

Theorem 5.1 — Optimal Mean-Field Factor

Under the mean-field assumption, the ELBO-optimal jth factor satisfies:

q*_j(z_j) ∝ exp(𝔼_{q_{−j}}[log p(z, x)])

where the expectation is over all variables z_k (k ≠ j) under their current variational distributions.

CAVI: iterate this update for j = 1, …, J until convergence. Each step monotonically increases the ELBO (coordinate ascent in a jointly non-concave objective).

CAVI was the dominant VI algorithm for latent Dirichlet allocation (LDA), mixture models, and hierarchical Bayesian models throughout the 2000s. Its strength is the clean, closed-form updates available when the model is in the **conjugate exponential family**: if $p(z_j \mid z_{-j}, x)$ belongs to the exponential family, the optimal q_j^* has the same form and the natural parameters update as linear functions of expectations under other factors.

5.3 Black-Box Variational Inference (BBVI)

CAVI requires closed-form coordinate updates unavailable for non-conjugate models. **Black-box VI** (Ranganath et al. 2014) estimates the ELBO gradient via Monte Carlo sampling. The gradient with respect to φ:

$$\nabla_\varphi \mathscr{L} = \mathbb{E}_{q_\varphi}[\nabla_\varphi \log q_\varphi(z) \times (\log p(x, z) - \log q_\varphi(z))]$$

(the **score function estimator** or REINFORCE gradient). This estimator is unbiased but has extremely high variance: the term $\log p(x,z) - \log q_\varphi(z)$ (the "control variate") must be near zero for low variance, but there is no natural baseline without model-specific information.

The **reparametrisation trick** provides a low-variance alternative when q_φ admits a differentiable reparametrisation $z = g(\varepsilon; \varphi)$ with $\varepsilon \sim \mathcal{N}(0,I)$:

$$\nabla_\varphi \mathscr{L} = \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,I)}[\nabla_\varphi \log p(x, g(\varepsilon;\varphi)) - \nabla_\varphi \log q_\varphi(g(\varepsilon;\varphi))]$$

This gradient flows through the sampler via the Jacobian $\partial g/\partial\varphi$, giving much lower variance. It requires that q_φ be reparametrisable — satisfied by Gaussians (location-scale family), Gamma, Beta, and many other distributions, but not discrete distributions. The reparametrisation trick is the cornerstone of the VAE and of modern stochastic variational inference.

Theorem 5.2 — Reparametrisation Gradient Variance

For the reparametrisation estimator ĝ_rep and score function estimator ĝ_score:

Var[ĝ_rep] = O(‖J‖²_F × Var_ε[log p gradient])

Var[ĝ_score] = Var_z[(log p − log q) × score]

The reparametrisation estimator is typically orders of magnitude lower variance because it exploits the smoothness of p with respect to φ, while the score estimator treats the model as a black box.

5.4 Importance-Weighted Autoencoder (IWAE)

The ELBO uses a single-sample Monte Carlo estimate of $\log p(x)$. The **Importance-Weighted Autoencoder** (Burda et al. 2015) tightens this bound by averaging over K importance-weighted samples:

$$\mathcal{L}_K(\theta, \varphi; x) = \mathbb{E}_{\{z_1,\ldots,z_K \sim q_\varphi(\cdot|x)\}}[\log (1/K) \Sigma_k w_k]$$

$$\text{where } w_k = p_\theta(x, z_k) / q_\varphi(z_k|x) \text{ (unnormalised importance weights)}$$

Theorem 5.3 — IWAE Bound Hierarchy

The bounds satisfy: $\log p_\theta(x) \geq \mathcal{L}_{K+1} \geq \mathcal{L}_K \geq \mathcal{L}_1 = \text{ELBO}$, with equality as $K \to \infty$.

Proof: By Jensen's inequality applied to the concave log function:

$\mathcal{L}_K = \mathbb{E}[\log (1/K)\Sigma w_k] \leq \log \mathbb{E}[(1/K)\Sigma w_k] = \log p_\theta(x)$

The gap $\mathcal{L}_\infty - \mathcal{L}_K = O(\text{Var}[w]/K)$ by the delta method, measuring importance weight degeneracy.

The IWAE gradient has a crucial pathology: for large K, the gradient signal is dominated by the single sample with the highest weight, causing φ to update almost exclusively based on high-probability regions of $p_\theta(z|x)$. This **posterior collapse** problem means that tighter bounds do not necessarily lead to better representation learning. The IWAE-DReG (Doubly-Reparametrised Gradient) estimator corrects this by stopping gradients through the weight normalisation:

$$\nabla_\varphi \mathcal{L}_K^{DReG} = \mathbb{E}[\Sigma_k \tilde{w}_k^2 \nabla_\varphi \log q_\varphi(z_k|x)] \quad \text{where } \tilde{w}_k = w_k / \Sigma_j w_j \text{ (stopped)}$$

5.5 Normalising Flows for Variational Inference

Mean-field variational inference is limited by the expressiveness of the factored Gaussian posterior. **Normalising flows** enrich the variational family by composing a

sequence of invertible transformations $z_K = f_K \circ \ldots \circ f_1(z_0)$ starting from a simple distribution $z_0 \sim q_0$:

$$\log q_K(z_K) = \log q_0(z_0) - \Sigma_t \log |\det J_t|$$

The ELBO with flow-based posterior:

$$\mathscr{L} = \mathbb{E}_{q_0}[\log p(x, z_K) - \log q_0(z_0) + \Sigma_t \log |\det J_t|]$$

Common flow architectures for VI include **IAF** (Inverse Autoregressive Flows), which achieve efficient sampling but slow density evaluation; **Planar Flows** with transformations $f(z) = z + uh(w^T z + b)$ (one-rank perturbations); and **Radial Flows** expanding/contracting around reference points. The expressiveness-computation tradeoff determines which flows are practical for VI: sampling (for the ELBO) should be fast, while log-density evaluation (for gradient computation) must also be tractable.

Chapter 6: Uncertainty Quantification

A well-calibrated predictive system should know what it does not know. Uncertainty quantification (UQ) methods transform a point-prediction neural network into a system that assigns reliable confidence levels to its outputs — confidence that tracks true predictive uncertainty rather than being an artefact of the training objective or architecture. The methods range from principled Bayesian approaches to practical ensemble and conformal methods.

6.1 Bayesian Neural Networks

A **Bayesian neural network** (BNN) places a prior $p(\omega)$ over network weights ω and performs inference via the posterior $p(\omega \mid \mathcal{D}) \propto p(\omega)p(\mathcal{D} \mid \omega)$. Predictions are marginalised over the posterior:

$$p(y^* \mid x^*, \mathcal{D}) = \int p(y^* \mid x^*, \omega)\, p(\omega \mid \mathcal{D})\, d\omega$$

This integral is intractable for neural networks. The key challenges are: (1) the posterior $p(\omega \mid \mathcal{D})$ is high-dimensional, multimodal, and highly non-Gaussian; (2) standard MCMC (NUTS) does not scale to millions of parameters; (3) mean-field VI underestimates posterior uncertainty because the factored Gaussian approximation is too simple. SGLD addresses (2) at the cost of exact samples; structured VI (including matrix-Gaussian posteriors) partially addresses (3).

The practical utility of BNNs lies in the decomposition of predictive uncertainty into **aleatoric uncertainty** (irreducible noise in the data: $\mathbb{E}_\omega[\mathrm{Var}_{y \mid x,\omega}[y]]$) and **epistemic uncertainty** (model uncertainty reducible with more data: $\mathrm{Var}_\omega[\mathbb{E}_{y \mid x,\omega}[y]]$). For out-of-distribution (OOD) inputs, epistemic uncertainty should be high — a well-calibrated BNN warns when it is being asked to extrapolate beyond training data.

6.2 Laplace Approximation

The Laplace approximation fits a Gaussian to the posterior at the MAP estimate. Given $\hat{\omega} = \mathrm{argmax}\, \log p(\omega \mid \mathcal{D})$:

$$p(\omega \mid \mathcal{D}) \approx \mathcal{N}(\hat{\omega}, H^{-1}) \quad \text{where} \quad H = -\nabla^2_{\omega} \log p(\omega \mid \mathcal{D})\big|_{\omega=\hat{\omega}}$$

The Hessian H at the optimum is positive definite (MAP is a maximum), giving a valid Gaussian approximation. Predictive distribution:

$$p(y^* \mid x^*, \mathcal{D}) \approx \int p(y^* \mid x^*, \omega)\, \mathcal{N}(\omega;\, \hat{\omega}, H^{-1})\, d\omega$$

For neural networks with millions of parameters, computing and inverting the full Hessian is infeasible. Kronecker-Factored Laplace (**KFAC-Laplace**) approximates the Hessian block-diagonally per layer as a Kronecker product $H^{(l)} \approx A^{(l)} \otimes S^{(l)}$ — the same factorisation as K-FAC. Diagonal Laplace (**DiagLaplace**) uses only the diagonal of H, equivalent to independent Gaussian uncertainty per weight.

Theorem 6.1 — Laplace Approximation Quality

The Laplace approximation is exact at the optimum in the limit of infinite data (posterior concentrates at MAP). The first-order error term is:

$$D_{KL}(p_{Laplace} \| p_{posterior}) = O(\|\nabla^3 \log p\| / n^{1/2})$$

The approximation fails when: (i) the MAP is not the global mode, (ii) the posterior is multimodal, (iii) the curvature at the MAP does not represent the global geometry.

For OOD detection: the Laplace predictive variance is large when the test point lies in a region where the linearised network features are dissimilar from training data.

6.3 Deep Ensembles

Deep Ensembles (Lakshminarayanan et al. 2017) trains M independently initialised and trained networks and combines their predictions:

$$\bar{p}(y^* \mid x^*) = (1/M) \sum_m p_{\theta_m}(y^* \mid x^*)$$

For a Gaussian output head, the ensemble mean and variance are:

$$\bar{\mu} = (1/M) \sum \mu_m(x^*), \quad \bar{\sigma}^2 = (1/M) \sum(\sigma_m^2(x^*) + \mu_m(x^*)^2) - \bar{\mu}^2$$

The second term is the **epistemic uncertainty** from disagreement between ensemble members. Despite having no Bayesian justification, deep ensembles are empirically the strongest known UQ method — consistently outperforming BNNs, Laplace approximations, and MC Dropout across benchmarks. The likely explanation: the SGD loss landscape has multiple distinct good minima corresponding to qualitatively different function representations; independent training initialises near different basins, and the ensemble captures genuine diversity.

Theorem 6.2 — Ensemble Variance as Epistemic Uncertainty

For M independent models $\theta_1, \ldots, \theta_M \sim p(\theta \mid \circledcirc)$, the ensemble variance equals the posterior predictive variance:

$$\mathrm{Var}[\bar{y}] = \mathbb{E}[\mathrm{Var}[y \mid x, \theta]] + \mathrm{Var}[\mathbb{E}[y \mid x, \theta]] \quad \text{(bias-variance decomposition)}$$

The first term is aleatoric; the second is epistemic. With independent training, the M models approximate independent posterior samples — empirically confirmed by comparing ensemble calibration to approximate Bayes.

6.4 Conformal Prediction

Conformal Prediction (Vovk, Gammerman, Shafer 1999) provides distribution-free, finite-sample coverage guarantees for predictive sets without any assumptions on the data distribution beyond exchangeability. For a user-specified confidence level $1-\alpha$, conformal prediction produces a set $C(x_{n+1})$ that contains the true label with probability at least $1-\alpha$:

$$P(y_{n+1} \in C(x_{n+1})) \geq 1 - \alpha$$

Theorem 6.3 — Conformal Coverage Guarantee

Let $\{(x_1,y_1),\ldots,(x_n,y_n), (x_{n+1},y_{n+1})\}$ be exchangeable. For any conformity score $s(x,y)$:

Let $\hat{q} = (\lceil (n+1)(1-\alpha) \rceil / n)$-quantile of $\{s(x_i,y_i)\}_{i=1}^{n}$

Then $C(x_{n+1}) = \{y : s(x_{n+1}, y) \leq \hat{q}\}$ satisfies $P(y_{n+1} \in C(x)) \geq 1-\alpha$.

The guarantee is exact: $P(y_{n+1} \in C) = \lceil (n+1)(1-\alpha) \rceil / (n+1)$ — never more conservative than necessary.

The conformity score $s(x, y)$ is a non-conformity measure: for regression, $s(x,y) = |y - f(x)|$; for classification with softmax output, $s(x,y) = 1 - p_f(y|x)$. **Split conformal** divides data into calibration and test sets — the simplest and most practical variant. **Full conformal** retrains for each test point (exact but expensive). **Mondrian conformal** provides class-conditional coverage by computing quantiles separately per class.

Conformal risk control (Angelopoulos et al. 2022) generalises the coverage guarantee to arbitrary monotone risk functionals: for any user-specified risk R (error rate, false negative rate, expected set size), one can calibrate the prediction set to satisfy $\mathbb{E}[R(C(x_{n+1}))] \leq \alpha$. This extends conformal coverage to structured prediction (segmentation masks, graph outputs) and complex risk measures.

6.5 Calibration Assessment and the Reliability Diagram

A classifier is **perfectly calibrated** if among all instances where it predicts confidence p, exactly fraction p are correct:

$$P(y = \hat{y} \mid \hat{p}(\hat{y} \mid x) = p) = p \quad \text{for all } p \in [0,1]$$

The **Expected Calibration Error** (ECE) estimates this empirically by binning predictions into M equal-width intervals and computing the weighted average gap:

$$ECE = \Sigma_m \, |B_m|/n \times |acc(B_m) - conf(B_m)|$$

where $acc(B_m) = correct_in_bin \, / \, |B_m|$ and $conf(B_m) = mean(\hat{p})$ in bin m. Standard deep networks trained with cross-entropy loss are systematically overconfident: predicted confidence is higher than empirical accuracy. **Temperature scaling** (dividing logits by a scalar $T > 1$ before softmax) is the simplest post-hoc calibration: one parameter adjusted on a held-out calibration set, consistently outperforming more complex methods. The theoretical reason: T effectively broadens the predictive distribution, trading accuracy for calibration.

UQ Method	Coverage Guarantee	Scalability	Key Assumption	OOD Detection
Full BNN (NUTS)	Exact Bayes	Very low (d < 1000)	Prior specification	Excellent
SGLD / SGHMC	Approximate Bayes	Medium (millions)	Ergodicity + step size	Good
Laplace (KFAC)	Quadratic approx	High	Unimodal posterior	Moderate
MC Dropout	Approx. VI	High	Dropout ↔️ GP prior	Moderate
Deep Ensembles	None (empirical)	Medium (M models)	Independent minima	Excellent
Conformal	Distribution-free	Very high	Exchangeability only	Via coverage gap
Temperature Scaling	None (post-hoc)	Highest	Single scalar bias	None directly

PART IV

THE GENERATIVE PARADIGM

Variational Autoencoders · GANs · Normalising Flows · Score-Based Models · Diffusion · Flow Matching · Latent Diffusion · EBMs · Self-Supervised Learning

Preface: The Problem of Generation

To generate is, in a precise mathematical sense, to sample from a probability distribution over a structured, high-dimensional space. The challenge is that such distributions — over natural images, molecular structures, natural language utterances — are supported on low-dimensional manifolds embedded in spaces of dimension $D = 10^6$ or more, and are accessible only through finite samples without any analytic description. This part of the treatise examines the mathematical architectures that have transformed this challenge into practical reality.

The progression from Variational Autoencoders to Generative Adversarial Networks to Normalising Flows to Score-Based Diffusion Models and Flow Matching is not merely chronological. It represents a sequence of fundamentally different mathematical answers — each generation revealing a new geometric or probabilistic structure, each resolving pathologies of its predecessor at the cost of introducing new ones. Understanding this progression is essential for understanding why contemporary generative systems work and what their remaining limitations are.

The unifying mathematical thread is **optimal transport**: the problem of finding the most efficient coupling between a simple reference distribution and the complex target. Diffusion models learn this transport via a stochastic denoising path; flow matching via a deterministic vector field; GANs adversarially; VAEs via variational lower bounds. The Wasserstein distance, the Benamou-Brenier dynamic formulation, and the Monge-Kantorovich problem are the backbone unifying these diverse approaches.

Chapter 1: Variational Autoencoders

The Variational Autoencoder (Kingma and Welling, 2013) is simultaneously a deep latent variable model, an amortised variational inference algorithm, and a representation learning system. Its mathematical elegance lies in the joint treatment of generation and inference within a single end-to-end differentiable framework governed by a single unified objective, the Evidence Lower Bound.

1.1 The Latent Variable Model and the ELBO

Let $z \in \mathbb{R}^d$ be a latent variable with prior $p(z) = \mathcal{N}(0, I^d)$ and let $x \in \mathbb{R}^D$ be observed, modelled by a decoder network $p_\theta(x|z)$. The joint is $p_\theta(x, z) = p_\theta(x|z)p(z)$ and the marginal $p_\theta(x) = \int p_\theta(x|z)p(z)\, dz$ is intractable. Introducing an approximate posterior $q_\varphi(z|x)$ and applying Jensen's inequality yields the ELBO:

$$\log p_\theta(x) \geq \mathcal{L}(\theta,\varphi;x) = \mathbb{E}_{q_\varphi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\varphi(z|x)\|p(z))$$

The first term is the **reconstruction term**: the expected log-likelihood of x given the latent encoding. The second term is the **regularisation term**: it penalises posteriors that deviate from the prior, preventing the encoder from simply memorising training points. For a diagonal Gaussian encoder $q_\varphi(z|x) = \mathcal{N}(\mu_\varphi(x), \mathrm{diag}(\sigma^2_\varphi(x)))$ and standard normal prior, the KL term has the closed form:

$$D_{KL}(\mathcal{N}(\mu, \sigma^2 I) \,\|\, \mathcal{N}(0, I)) = \tfrac{1}{2} \sum_j (\mu_j^2 + \sigma_j^2 - \log \sigma_j^2 - 1)$$

1.2 The Reparametrisation Trick

Optimising $\mathcal{L}$ with respect to φ requires backpropagating through the sampling operation $z \sim q_\varphi(z|x)$, which is non-differentiable. The

reparametrisation trick resolves this by writing $z = \mu_\varphi(x) + \sigma_\varphi(x) \odot \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, I)$, relocating the randomness to a fixed distribution:

$$\nabla_\varphi \mathbb{E}_{q_\varphi}[f(z)] = \mathbb{E}_\varepsilon[\nabla_\varphi f(\mu_\varphi(x) + \sigma_\varphi(x) \odot \varepsilon)]$$

This transformation produces unbiased gradient estimates with variance polynomial in the magnitude of the gradient of f, orders of magnitude lower than the score function (REINFORCE) estimator. The reparametrisation trick is the central enabling mechanism for the VAE and, more broadly, for the entire class of continuous latent variable models trained by amortised variational inference.

Theorem 1.1 (Rate-Distortion Interpretation of the ELBO)

The ELBO decomposes as: $\mathbb{E} = -$[distortion] − rate, where distortion = $-\log p_\theta(x|z)$ is the reconstruction error and rate = $D_{KL}(q_\varphi(z|x) \| p(z))$ measures the number of bits used by the latent code. The β-VAE generalises this to $\mathcal{L}_\beta = -\mathbb{E}[\text{distortion}] - \beta \cdot \text{rate}$. $\beta > 1$ encourages disentangled representations by more strongly penalising the rate; the tradeoff is fundamental — high decoder expressiveness drives the rate to zero (posterior collapse), while high β compresses representations at the cost of reconstruction quality.

1.3 Posterior Collapse and Hierarchical VAEs

Posterior collapse occurs when the decoder $p_\theta(x|z)$ becomes powerful enough to model x without reference to z. The posterior $q_\varphi(z|x)$ collapses to the prior and the KL term vanishes, producing a local ELBO maximum with zero mutual information between z and x. This is not a training failure in the standard sense — the ELBO is being maximised — but a degenerate solution where the latent code carries no information.

Hierarchical VAEs address posterior collapse by stacking multiple latent layers $z_L \to \dots \to z_1 \to x$. The ELBO telescopes as $\mathcal{L} = \mathbb{E}[\log p(x|z_1)] - \Sigma_l D_{KL}(q_\varphi(z_l|z_{\ell-1},x) \| p_\theta(z_l|z_{\ell-1}))$. NVAE, VDVAE, and Diffusion VAE achieve state-of-the-art density estimation by combining hierarchical structure with residual encoder-decoder networks and careful normalisation. The **aggregated posterior** $q_\varphi(z) = (1/N)\Sigma_i q_\varphi(z|x_i)$ must match the prior for clean generation; the holes problem arises when regions of $p(z)$ not covered by any $q_\varphi(z|x_i)$ produce poor samples when decoded. WAE replaces the per-

example KL with an MMD regulariser on the aggregated posterior, directly addressing this mismatch.

Chapter 2: Generative Adversarial Networks

The GAN framework (Goodfellow et al., 2014) transforms generation into a two-player zero-sum game, eliminating the need for explicit density estimation or variational bounds. A generator network learns to map noise to data; a discriminator learns to distinguish real from generated samples. The adversarial dynamic drives the generator distribution toward the data distribution without ever requiring access to that distribution's density.

2.1 The Minimax Game and JS Divergence

Given data distribution p_r and generator distribution $p_g = G_\#p_z$ (the pushforward of noise under G), the GAN objective is:

$$\min_G \max_D \; V(D,G) = \mathbb{E}_{x\sim p_r}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1-D(G(z)))]$$

Theorem 2.1 (GAN Optimal Discriminator and Global Minimum)

For fixed G, the optimal discriminator is $D^*_G(x) = p_r(x) / (p_r(x) + p_g(x))$. Substituting D^*_G yields the reduced objective $C(G) = -\log 4 + 2\cdot JSD(p_r \,\|\, p_g)$, where JSD is the Jensen-Shannon divergence. The global minimum $C(G^*) = -\log 4$ is attained iff $p_g = p_r$. However, when p_r and p_g have disjoint supports — typical when p_g is supported on a low-dimensional manifold — JSD = log 2 is constant and gradients vanish, creating the vanishing gradient pathology of the original GAN objective.

2.2 Wasserstein GAN and the Earth Mover Distance

WGAN (Arjovsky et al., 2017) replaces JS divergence with the **Wasserstein-1 distance**:

$$W(p_r, p_g) = \inf_{\gamma\in\Pi(p_r,p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|]$$

By the Kantorovich-Rubinstein duality, $W(p_r, p_g) = \sup_{\|f\|_L\le 1} \mathbb{E}_{x\sim p_r}[f(x)] - \mathbb{E}_{y\sim p_g}[f(y)]$ over 1-Lipschitz functions. WGAN-GP enforces the

Lipschitz constraint via a gradient penalty: $\mathscr{L}_{GP} = \mathbb{E}_{\hat{x}}[(\|\nabla_{\hat{x}} f_w(\hat{x})\|_2 - 1)^2]$ where $\hat{x}$ is sampled along interpolations between real and generated points.

Theorem 2.2 (WGAN Gradient Properties)

The Wasserstein distance satisfies: (i) W(p_r, p_g) is continuous everywhere and differentiable almost everywhere, even when supports are disjoint. (ii) ∇_θ W(p_r, p_{g_θ}) is always informative — the gradient does not vanish regardless of the discriminator quality. (iii) W correlates monotonically with sample quality empirically, unlike JSD which saturates to log 2 once supports separate. These properties make WGAN training substantially more stable than the original GAN.

2.3 StyleGAN, Progressive Growing, and Latent Disentanglement

StyleGAN (Karras et al., 2019) introduces an intermediate latent space W obtained from noise z via a learned mapping network $f: Z \to W$. Styles $w \in W$ are injected at each resolution level via Adaptive Instance Normalisation (AdaIN):

$$AdaIN(x_i, w) = y_{s,i} \cdot (x_i - \mu(x_i))/\sigma(x_i) + y_{b,i}, \quad (y_s, y_b) = A(w)$$

The W space achieves lower perceptual path length (PPL) than Z, indicating more disentangled interpolations where each direction corresponds to an independent semantic attribute. StyleGAN2 eliminates blob artefacts via revised normalisation; StyleGAN3 replaces the feature map with continuous signal representations to eliminate texture sticking, achieving true equivariance to translation and rotation in the synthesis network.

2.4 GAN Training Dynamics, Mode Collapse, and Spectral Normalisation

Mode collapse arises because the generator gradient $\nabla_\theta \mathbb{E}_z[-\log D(G_\theta(z))]$ pushes G toward regions where D assigns high probability, but if D concentrates mass on a few modes, G satisfies this constraint by covering only those modes. Formally, mode collapse is a non-uniform covering of the data manifold: $supp(p_g) \subsetneq supp(p_r)$.

Spectral normalisation (Miyato et al., 2018) enforces the 1-Lipschitz constraint of the discriminator by normalising each weight matrix by its spectral norm: $\hat{W} = W / \sigma_{max}(W)$, where $\sigma_{max}(W)$ is estimated by the power iteration method. This prevents the discriminator from becoming too confident while maintaining expressive capacity. The theoretical justification connects to the Lipschitz constant of the discriminator function: for a network with L layers, $\|f\|_{Lip} \leq \prod_l \sigma_{max}(W_l)$, so normalising each layer to unit spectral norm bounds the global Lipschitz constant by 1.

Chapter 3: Normalising Flows

Normalising flows construct an explicit bijection between a simple base distribution and the data distribution, enabling exact likelihood computation and exact sampling. Unlike VAEs (approximate posterior) and GANs (no likelihood), flows trade off architectural expressivity for mathematical exactness. Their theoretical foundation is the change-of-variables formula, which specifies how probability densities transform under smooth invertible maps.

3.1 The Change-of-Variables Formula

Let $z \sim p_z(z)$ be a base distribution and $f: \mathbb{R}^d \to \mathbb{R}^d$ a smooth bijection. The density of the pushforward $x = f(z)$ is:

$$\log p_x(x) = \log p_z(f^{-1}(x)) + \log \left| \det J_{f^{-1}}(x) \right|$$

where $J_{f^{-1}}$ is the Jacobian of f^{-1}. For a composition of flows $f = f_K \circ \ldots \circ f_1$, the log-likelihood decomposes as $\log p_x(x) = \log p_z(z_0) + \sum_k \log \left| \det J_{f_k^{-1}}(z_k) \right|$. The fundamental constraint for practical flows is that the Jacobian determinant must be computable in $O(d)$ rather than $O(d^3)$ (the cost of a general matrix determinant).

Theorem 3.1 (Triangular Jacobian and Efficient Determinant)

If $f: \mathbb{R}^d \to \mathbb{R}^d$ has a triangular Jacobian matrix J_f with diagonal entries $(\partial f_i / \partial z_i)$, then $\det(J_f) = \prod_i (\partial f_i / \partial z_i)$. This is computable in $O(d)$. Coupling layers exploit this by partitioning $z = (z_{1:d/2}, z_{d/2+1:d})$ and defining: $x_{1:d/2} = z_{1:d/2}$, $x_{d/2+1:d} = z_{d/2+1:d} \odot \exp(s(z_{1:d/2})) + t(z_{1:d/2})$. The Jacobian is block-triangular with the identity on the upper-left block, giving $\log |\det J| = \sum_i s_i(z_{1:d/2})$.

3.2 Coupling Layers: RealNVP and Glow

RealNVP (Dinh et al., 2017) implements affine coupling layers with alternating masks to ensure all dimensions are transformed. **Glow** (Kingma and Dhariwal, 2018) extends RealNVP with three components: **ActNorm** (affine normalisation with per-channel scale and bias initialised from the first minibatch), **1×1 invertible convolutions** (generalising channel-permutations; the log-determinant is $h \cdot w \cdot \log |\det W|$ for an $h \times w$ feature map), and **affine coupling layers**. Glow achieves high-fidelity image synthesis and, crucially, enables semantically meaningful latent space interpolations because the latent space is in exact bijection with the data space.

3.3 Autoregressive Flows and the IAF/MAF Duality

Autoregressive flows parameterise the transformation as $x_i = \mu_i(x_\{<i\}) + \sigma_i(x_\{<i\}) \cdot z_i$, where each output dimension depends on all previous output dimensions. The Jacobian is lower-triangular, giving efficient determinant computation. The key asymmetry is between **Masked Autoregressive Flow (MAF)** and **Inverse Autoregressive Flow (IAF)**:

MAF (Papamakarios et al., 2017): sampling requires D sequential passes (each x_i depends on x_{<i}, which are not yet computed); density evaluation is parallelisable. Efficient for training, slow for sampling.

IAF (Kingma et al., 2016): sampling is parallelisable (each z_i is immediately available from the prior); density evaluation requires D sequential passes. Efficient for sampling, slow for density evaluation. MAF and IAF are exact duals: the same computation graph, traversed in opposite directions.

3.4 Continuous Normalising Flows and FFJORD

Continuous Normalising Flows (CNFs) replace the finite sequence of transformations with a continuous ODE: $dz/dt = f_\theta(z(t), t)$. The instantaneous change in log-density is:

$$d \log p(z(t))/dt = -Tr(\partial f_\theta/\partial z(t))$$

FFJORD (Grathwohl et al., 2019) estimates the trace using the **Hutchinson trace estimator**: $Tr(A) = \mathbb{E}_\{\varepsilon\}[\varepsilon^T A \varepsilon]$ for $\varepsilon \sim \mathcal{N}(0,I)$ or Rademacher distribution.

This reduces the cost of the log-determinant from $O(d^2)$ to $O(d)$ per function evaluation. CNFs achieve state-of-the-art density estimation on tabular data and are the theoretical precursor to Flow Matching (Chapter 7), which replaces the ODE-based training with direct regression on conditional vector fields.

Chapter 4: Score-Based Models and Stochastic Differential Equations

Score-based generative models represent a profound reconceptualisation of the generation problem: rather than learning to transform a simple distribution into a complex one directly, they learn the gradient of the log-density (the score function) at every noise level, then use this learned score to reverse a known noising process. The mathematical framework connecting SMLD, DDPM, and modern diffusion models is Song et al.'s (2021) unification via stochastic differential equations.

4.1 Score Matching

Let $p_data(x)$ be the data distribution. The score function is $s^*(x) = \nabla_x \log p_data(x)$. Score matching (Hyvärinen, 2005) proposes to learn $s_\theta(x) \approx s^*(x)$ by minimising the Fisher divergence:

$$J(\theta) = \mathbb{E}_{p_data}[\tfrac{1}{2}\|s_\theta(x)\|^2 + Tr(\nabla_x s_\theta(x))]$$

The trace term $Tr(\nabla_x s_\theta(x))$ is the divergence of the score network, costly to compute exactly. Sliced score matching uses random projections to estimate it. **Denoising score matching** (Vincent, 2011) bypasses this by perturbing data with noise $\tilde{x} = x + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ and training:

$$J_DSM(\theta) = \mathbb{E}_{x,\varepsilon}[\|s_\theta(\tilde{x}, \sigma) - \nabla_{\tilde{x}} \log p(\tilde{x}|x)\|^2] = \mathbb{E}_{x,\varepsilon}[\|s_\theta(\tilde{x}, \sigma) + \varepsilon/\sigma^2\|^2]$$

Denoising score matching admits an unbiased estimator without trace computation and forms the training objective for all practical diffusion models.

4.2 The SDE Framework: Unifying SMLD and DDPM

Song et al. (2021) show that both SMLD (adding Gaussian noise of increasing variance) and DDPM (Gaussian diffusion) are discretisations of SDEs. The general **forward SDE** is:

$$dx = f(x,\, t)dt + g(t)dW$$

where f is the drift coefficient and g the diffusion coefficient. The corresponding **reverse SDE** (Anderson, 1982) is:

$$dx = [f(x,t) - g(t)^2\, \nabla_x \log p_t(x)]dt + g(t)d\bar{W}$$

where $d\bar{W}$ is reverse-time Brownian motion. If the score $\nabla_x \log p_t(x)$ is known, samples from p_data can be obtained by solving the reverse SDE from $t=T$ (where p_T is approximately Gaussian) to $t=0$. The VPSDE (Variance-Preserving SDE, corresponding to DDPM) uses $f(x,t) = -\frac{1}{2}\beta(t)x$ and $g(t) = \sqrt{\beta(t)}$; the VESDE (Variance-Exploding, corresponding to SMLD) uses $f=0$ and $g(t) = d\sigma(t)/dt$

Theorem 4.1 (Probability Flow ODE)

For any SDE of the form dx = f(x,t)dt + g(t)dW, there exists a deterministic ODE — the probability flow ODE — with the same marginal distributions $\{p_t\}$: $dx/dt = f(x,t) - \frac{1}{2}g(t)^2\, \nabla_x \log p_t(x)$This ODE can be solved with any black-box ODE solver and enables exact likelihood computation via the instantaneous change-of-variables formula. Empirically, DPM-Solver and DEIS achieve high-quality samples in 10–20 NFEs (neural function evaluations) by exploiting the semi-linear structure of the probability flow ODE and caching evaluations across steps.

Chapter 5: DDPM, DDIM, and Classifier-Free Guidance

Denoising Diffusion Probabilistic Models (Ho et al., 2020) provide a discrete-time instantiation of the SDE framework that connects score matching to variational inference. Their success derives from three contributions: the derivation of the denoising objective as a variational lower bound, the parametrisation of the network to predict noise rather than score, and the U-Net architecture that processes multi-scale features.

5.1 DDPM: The Variational Lower Bound

The forward process adds Gaussian noise over T steps: $q(x_t \mid x_\{t-1\}) = \mathcal{N}(x_t;\, \sqrt{(1-\beta_t)}x_\{t-1\},\, \beta_t I)$. The marginal has a convenient closed form:

$$q(x_t \mid x_0) = \mathcal{N}(x_t;\, \sqrt{\bar{\alpha}_t}\, x_0,\, (1-\bar{\alpha}_t)I), \quad \bar{\alpha}_t = \Pi_{s=1}^{t}(1-\beta_s)$$

The reverse process $p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$ is trained by maximising the ELBO. Ho et al. show that parameterising μ_θ in terms of a predicted noise $\varepsilon_\theta(x_t, t)$ and fixing $\Sigma_\theta = \sigma_t^2 I$ yields the simplified objective:

$$\mathcal{L}_{simple}(\theta) = \mathbb{E}_{t, x_0, \varepsilon}[\|\varepsilon - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon,\, t)\|^2]$$

This objective is equivalent to denoising score matching at each noise level, with the weighting determined by the noise schedule. The result is a model that learns to denoise at all noise levels simultaneously, with each level corresponding to a different scale of structure in the data.

5.2 DDIM: Deterministic Sampling and the ODE Connection

DDIM (Song et al., 2021) derives a non-Markovian inference process consistent with the same marginals as DDPM but allowing deterministic sampling. The DDIM update is:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \cdot \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t} \cdot \varepsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \cdot \varepsilon_\theta(x_t, t) + \sigma_t \cdot \varepsilon$$

Setting $\sigma_t = 0$ gives a deterministic trajectory — the same sample is produced from the same initial noise, enabling **latent space interpolation** by interpolating starting noises. The deterministic DDIM corresponds precisely to the Euler discretisation of the probability flow ODE, bridging the discrete and continuous perspectives. DDIM achieves comparable quality to DDPM in 50–100 steps versus DDPM's 1000, a 10–20× speedup.

5.3 Classifier-Free Guidance

Classifier guidance (Dhariwal and Nichol, 2021) steers the reverse diffusion process toward a class y by augmenting the score with the gradient of a classifier: $\nabla_x \log p(x_t \mid y) = \nabla_x \log p_t(x_t) + \gamma \nabla_x \log p(y \mid x_t)$. **Classifier-free guidance** (Ho and Salimans, 2021) eliminates the separate classifier by jointly training conditional and unconditional models and at inference combining:

$$\tilde{\varepsilon}_\theta(x_t, t, y) = \varepsilon_\theta(x_t, t, \varnothing) + \gamma \cdot (\varepsilon_\theta(x_t, t, y) - \varepsilon_\theta(x_t, t, \varnothing))$$

The guidance scale $\gamma > 1$ amplifies the conditional signal, trading diversity for fidelity and condition adherence. The theoretical interpretation is that classifier-free guidance samples from the distribution $p(x \mid y)^\gamma / Z$, a sharpened version of the conditional distribution that concentrates around its modes. This mechanism is central to text-to-image systems (Stable Diffusion, DALL-E 2,

Imagen) and underlies modern language model alignment via the connection between guidance and reward-weighted regression.

Chapter 6: Consistency Models

Consistency models (Song et al., 2023) address the fundamental tension in diffusion models between sample quality and sampling speed. Diffusion requires many sequential neural function evaluations to trace the reverse SDE; consistency models learn to map any point on the PF ODE trajectory directly to the trajectory's origin, enabling one-step generation while maintaining quality competitive with multi-step diffusion.

6.1 The Consistency Function

The **consistency function** $f: (x_t, t) \mapsto x_0$ maps any noisy point x_t on the PF ODE trajectory to the trajectory's starting point x_0. It must satisfy the **self-consistency property**: for any two times t_1, t_2 on the same trajectory,

$$f(x_{t1}, t_1) = f(x_{t2}, t_2)$$

The boundary condition $f(x_\varepsilon, \varepsilon) = x_\varepsilon$ (for small ε) is enforced by architecture (e.g., skip connections that bypass the network for small t). The consistency distillation (CD) objective trains f_θ to agree with $f_\{\theta^-\}$ (an exponential moving average target network) on adjacent trajectory points:

$$\mathscr{L}_CD(\theta) = \mathbb{E}[d(f_\theta(x_{tn+1}, t_{n+1}), f_\{\theta^-\}(\hat{x}_{tn}, t_n))]$$

where $\hat{x}_{tn}$ is obtained by one step of the ODE solver from x_{tn+1}. Consistency training (CT) removes the dependency on a pre-trained diffusion model by deriving the target from a weighted first-order Taylor expansion of the self-consistency condition.

Chapter 7: Flow Matching and Optimal Transport

Flow Matching (Lipman et al., 2022; Liu et al., 2022) provides a conceptually clean alternative to score-based diffusion: instead of learning to reverse a noising SDE, it directly regresses on a vector field that pushes a simple distribution toward the data distribution. This formulation eliminates the need for the Ito calculus of SDEs and often leads to straighter, more efficient sampling trajectories.

7.1 Conditional Flow Matching

Let $p_0 = \mathcal{N}(0,I)$ and $p_1 = p_data$. Flow matching seeks a vector field $u_t: \mathbb{R}^d \times [0,1] \to \mathbb{R}^d$ whose generated flow φ_t satisfies $[\varphi_t]_\#p_0 = p_t$ for all t. The **marginal flow matching loss** is:

$$\mathscr{L}_FM(\theta) = \mathbb{E}_{t,x\sim p_t}[\|v_\theta(x_t, t) - u_t(x_t)\|^2]$$

Since the marginal vector field u_t is intractable, Conditional Flow Matching (CFM) conditions on the data point x_1: $u_t(x \mid x_1)$ defines a simple path from $x_0 \sim p_0$ to x_1. The CFM objective is:

$$\mathscr{L}_CFM(\theta) = \mathbb{E}_{t,x_0,x_1}[\|v_\theta(\varphi_t(x_0,x_1), t) - (x_1 - x_0)\|^2]$$

for the simple linear interpolant $\varphi_t(x_0,x_1) = (1-t)x_0 + tx_1$. The key theorem is that CFM and FM have identical gradients despite conditioning on x_1, so CFM is an efficient estimator of FM despite using a simpler, tractable objective.

7.2 OT-Flow Matching and Straight Trajectories

The choice of coupling between $x_0 \sim p_0$ and $x_1 \sim p_data$ is not fixed — any coupling $\pi \in \Pi(p_0, p_data)$ induces a valid flow. The **optimal transport coupling** $\pi^* = \operatorname{argmin}_{\pi} \mathbb{E}_{(x_0,x_1)\sim\pi}[\|x_0 - x_1\|^2]$ minimises the expected transport cost, producing straighter flow trajectories that require fewer ODE solver steps. OT-FM (Lipman et al., 2022) uses the Minibatch OT coupling, tractable for finite batch sizes.

Theorem 7.1 (Straightness and NFE Efficiency)

For a flow generated by interpolant $\varphi_t(x_0,x_1)$, define the straightness as $S = 1 - \mathbb{E}[\|\varphi_t(x_0,x_1) - (t\cdot x_1 + (1-t)\cdot x_0)\|^2] / \mathbb{E}[\|x_1 - x_0\|^2]$. The OT coupling maximises straightness among all couplings with the same marginals. Straight flows require fewer steps for accurate ODE integration: a flow with curvature κ requires $O(\kappa/\varepsilon)$ steps for error ε, while a straight flow requires $O(1/\varepsilon)$. This is the theoretical justification for why OT-FM achieves high quality in ≤20 steps while independent coupling FM may require 100+ steps.

7.3 Stochastic Interpolants and the Broader Framework

Stochastic interpolants (Albergo and Vanden-Eijnden, 2023) unify diffusion models and flow matching within a single framework parameterised by the interpolant $I_t(x_0,x_1) = \alpha(t)x_0 + \beta(t)x_1 + \gamma(t)\varepsilon$ where $\varepsilon \sim \mathcal{N}(0,I)$ is independent noise. Different choices of (α, β, γ) recover different models: setting $\gamma=0$ recovers deterministic flow matching; setting $\alpha(t) = \sqrt{\bar{\alpha}_t}$ and $\beta(t) = \sqrt{(1-\bar{\alpha}_t)}$ recovers DDPM-type diffusion. The framework makes explicit that diffusion and flow matching differ primarily in whether independent noise is added to the interpolant, i.e., in whether the generation trajectory is stochastic or deterministic.

Chapter 8: Latent Diffusion Models and VQ-VAE

Applying diffusion models directly in pixel space is computationally prohibitive for high-resolution images: the U-Net must process tensors of dimension 3×512×512 or larger at each of thousands of denoising steps. Latent Diffusion Models (Rombach et al., 2022) resolve this by decoupling the perceptual compression (removing high-frequency imperceptible details) from the generative modelling (learning the semantic structure of the compressed representations).

8.1 VQ-VAE: Vector Quantisation in Latent Space

The Vector Quantised VAE (van den Oord et al., 2017) learns a discrete latent representation. An encoder maps x to a continuous embedding z_e, which is then quantised to the nearest codebook entry: $z_q = argmin_\{e_k \in C\} \|z_e - e_k\|_2$. The decoder reconstructs from z_q. The loss is:

$$\mathcal{L}_VQVAE = \|x - D(z_q)\|^2 + \|sg[z_e] - z_q\|^2 + \beta\|z_e - sg[z_q]\|^2$$

where $sg[\cdot]$ is the stop-gradient operator. The first term is reconstruction; the second (codebook loss) updates the embedding vectors; the third (commitment loss) encourages the encoder to commit to a codebook entry. The straight-through gradient $\nabla_\{z_e\}\mathcal{L} \approx \nabla_\{z_q\}\mathcal{L}$ allows approximate gradient flow through the non-differentiable argmin operation. VQ-VAE-2 extends this with hierarchical codebooks at multiple resolutions, enabling high-fidelity image synthesis via autoregressive modelling over the discrete codes.

8.2 Latent Diffusion Models and Cross-Attention Conditioning

LDM (Rombach et al., 2022) trains a KL-regularised or VQ-regularised autoencoder $E: X \to Z$ and $D: Z \to X$ to compress images by a factor of 4–8 in each spatial dimension. A diffusion model is then trained on the compressed latent code $z = E(x)$. The computational savings are f^4 for an f-fold spatial compression (since attention scales quadratically with spatial resolution). Conditioning on text or other modalities is incorporated via **cross-attention**:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(QK^T/\sqrt{d}) \cdot V, \quad Q = W_Q \cdot \varphi(z_t), \quad K = V = W_{K,V} \cdot \tau_\theta(y)$$

where $\tau_\theta(y)$ is a domain-specific encoder (CLIP text encoder, spectrogram encoder, etc.) and $\varphi(z_t)$ is a flattened spatial feature map. The cross-attention allows the denoising U-Net to attend to text tokens at every resolution, enabling text-guided synthesis (Stable Diffusion), inpainting, and super-resolution within the same architecture.

Chapter 9: Energy-Based Models and Modern Hopfield Networks

Energy-Based Models (LeCun et al., 2006) define a probability distribution over data through a scalar energy function rather than an explicit density. The framework is maximally general — any distribution can be represented — but training is challenging because the normalising constant (partition function) is intractable for flexible energy functions.

9.1 The EBM Framework and Contrastive Divergence

An EBM defines $p_\theta(x) = \exp(-E_\theta(x)) / Z_\theta$ where $Z_\theta = \int \exp(-E_\theta(x))\, dx$ is the partition function. The maximum likelihood gradient requires computing $\nabla_\theta \log p_\theta(x) = -\nabla_\theta E_\theta(x) + \mathbb{E}_{p_\theta}[\nabla_\theta E_\theta(x)]$. The second term — the expectation under the model — requires sampling from p_θ, typically via MCMC. **Contrastive divergence** (Hinton, 2002) approximates the model expectation by running MCMC for only k steps from data (CD-k), giving a biased but computationally efficient gradient estimate.

9.2 Modern Hopfield Networks and the Attention Connection

Classical Hopfield networks (1982) store memory patterns $\{\xi^u\}$ as attractors of the energy $E(x) = -\tfrac{1}{2} x^T W x$ with $W = \Sigma^u \xi^u \xi^{uT}$. Memory capacity is limited to

$O(n)$ patterns for n-dimensional states. Modern Hopfield Networks (Ramsauer et al., 2021) replace the quadratic energy with an exponential interaction:

$$E(x, \xi) = -lse(\beta, \Xi^T x) + \tfrac{1}{2}x^T x + \beta^{-1} \log M + \tfrac{1}{2}\beta^{-1} \max_\mu \|\xi^u\|^2$$

where $lse(\beta, z) = \beta^{-1} \log \Sigma_i \exp(\beta z_i)$ is the log-sum-exp. The update rule $x^{new} = \Xi \, softmax(\beta \Xi^T x)$ is precisely the attention mechanism: x plays the role of the query and Ξ plays the role of the keys and values. This connection reveals that transformer attention layers can be interpreted as one step of a Modern Hopfield update, and that the exponential energy gives the network capacity $2^{n/2}$ patterns — exponential in dimension — versus $O(n)$ for classical Hopfield.

Theorem 9.1 (Exponential Storage Capacity of Modern Hopfield Networks)

Let $\{\xi^u\}^M_{\mu=1}$ be M patterns with $\|\xi^u\| \leq 1$. The Modern Hopfield network with interaction function lse(β, ·) retrieves patterns with exponentially small error provided $M \leq \exp(\alpha n)$ for some $\alpha < \tfrac{1}{2}\beta^2 \cdot (\text{min separation})^2$. The retrieval error is bounded by $\exp(-\beta^2 \cdot (\text{separation})^2/2) \cdot n$. This exponential capacity is the key advantage over classical Hopfield: for n=512-dimensional embeddings, capacity scales as $\exp(256 \cdot \beta^2 \cdot \Delta^2)$ where Δ is the minimum separation between patterns, enabling transformer heads to function as associative memories for very large sets of key-value pairs.

Chapter 10: Self-Supervised Learning

Self-supervised learning aims to learn useful representations from unlabelled data by defining a pretext task whose labels are derived automatically from the data structure. The mathematical question is: which self-supervised objectives produce representations with provably good downstream transfer? The answer, developed through contrastive learning, distillation-based methods, and information-theoretic approaches, reveals a deep connection between self-supervised objectives and the geometry of feature spaces.

10.1 Contrastive Learning: SimCLR and the InfoNCE Objective

SimCLR (Chen et al., 2020) creates two augmented views x, x^+ of the same image and trains an encoder f plus projection head g to agree on their representations. The **NT-Xent (InfoNCE) loss** for a batch of N pairs:

$$\mathscr{L} = -(1/N)\,\Sigma_i \log \exp(\mathrm{sim}(z_i, z_i^+)/\tau) \,/\, [\Sigma_{j\neq i} \exp(\mathrm{sim}(z_i, z_j)/\tau)]$$

where $z_i = g(f(x_i))$ and $\mathrm{sim}(u,v) = u^T v/(\|u\|\|v\|)$ is cosine similarity. The theoretical justification (Wang and Isola, 2020) decomposes $\mathscr{L}$ into two terms: **alignment** (pushing positive pairs together) and **uniformity** (pushing the entire distribution toward uniform over the hypersphere), measured by the log of the average Gaussian kernel $\mathbb{E}_{(z,z')}[\exp(-2\|z-z'\|^2)]$. Optimal representations are uniform on the sphere while positive pairs are co-located — a precise geometric characterisation.

10.2 BYOL: Self-Distillation Without Negative Pairs

BYOL (Bootstrap Your Own Latent, Grill et al., 2020) trains an online network to predict the output of a target network (an exponential moving average of the online network) without negative pairs:

$$\mathscr{L}_{BYOL} = \|q_\theta(z_{online}) - sg[z_{target}]\|^2_2$$

The absence of negative pairs raises the collapse question: why does BYOL not collapse to constant representations? The key is the **asymmetry** between online (trained with gradients) and target (updated via EMA) networks, combined with batch normalisation in the predictor. Without either asymmetry, BYOL collapses. Rigorously, Tian et al. (2021) show that BatchNorm introduces an implicit negative pair mechanism via cross-batch statistics.

10.3 Barlow Twins and Information-Theoretic Redundancy Reduction

Barlow Twins (Zbontar et al., 2021) enforces decorrelation between feature dimensions by minimising the off-diagonal elements of the **cross-correlation matrix** between the two augmented-view representations:

$$\mathscr{L}_{BT} = \Sigma_i (1 - C_{ii})^2 + \lambda \Sigma_i \Sigma_{j\neq i} C_{ij}^2, \quad C_{ij} = \Sigma_b z_{i,b} z_{j,b} \,/\, (\|z_i\|\|z_j\|)$$

The first term is the **invariance term**: it pushes the diagonal toward 1, making the representation invariant to the augmentation. The second term is the **redundancy reduction term**: it pushes off-diagonal elements toward 0,

reducing statistical dependencies between feature dimensions. Barlow Twins does not require negative pairs, asymmetric networks, or stop-gradients, and the connection to **information theory** is explicit: the objective is a lower bound on the mutual information between the two views under the constraint that the feature distribution is approximately white (unit variance, zero correlation).

10.4 Masked Autoencoders

Masked Autoencoder (MAE, He et al., 2022) applies the BERT-style masking pretext task to visual data: randomly mask 75% of image patches and train a ViT encoder-decoder to reconstruct the masked pixels. The key design choices are: (1) **high masking ratio** (75% vs. BERT's 15%) creates a harder, more information-rich task; (2) the encoder operates only on **visible tokens** (no masking tokens in the encoder), dramatically reducing compute; (3) a shallow decoder processes the full sequence including mask tokens. MAE learns highly transferable representations: linear probing accuracy is within 1% of supervised pretraining, and fine-tuning surpasses it.

Conclusions: The Mathematical Unity of Generative Modelling

The generative paradigm reveals a profound mathematical unity underlying apparently disparate architectures. VAEs, GANs, flows, diffusion models, and consistency models are all solving the same fundamental problem — learning to transport a simple prior distribution to a complex data distribution — through different mathematical mechanisms that trade off different computational and statistical properties.

The connections between these methods are increasingly well-understood. Score matching is the continuous-time limit of denoising autoencoders. Diffusion models are continuous normalising flows trained via score matching. Flow matching generalises diffusion by replacing the SDE with an ODE and the score with a velocity field. Consistency models compress the multi-step ODE into a single-step map. The optimal transport coupling makes this map maximally straight, minimising the number of function evaluations.

What remains open is the theory of quality: why do diffusion models outperform VAEs for images despite the VAE's more principled probabilistic foundation? The answer involves the interplay between model capacity,

training dynamics, and the geometric structure of the data manifold — questions that the framework of stochastic interpolants and flow matching are beginning to address rigorously.

PART V

REINFORCEMENT LEARNING AND ALIGNMENT

MDP · Bellman · Policy Gradients · MaxEnt RL · Model-Based RL · RLHF · DPO · Constitutional AI · Emergent Properties · Mechanistic Interpretability

Introduction to Part V

Reinforcement learning addresses the problem of sequential decision-making under uncertainty: an agent interacts with an environment, receives scalar rewards, and must learn a policy that maximises cumulative return. Unlike supervised learning — where targets are provided — and generative modelling — where the data distribution is the object of interest — RL must deal with the fundamental challenge that actions affect future observations, creating temporal credit assignment problems and non-stationarity. This part develops the mathematical foundations of modern RL, from the Markov Decision Process formalism through deep RL algorithms and into the alignment problem: how to ensure that learned policies act in accordance with human values.

Chapter 1: Markov Decision Processes and the Bellman Equations

The Markov Decision Process provides the canonical mathematical abstraction of sequential decision-making. Its power comes from the Markov property: the

future is conditionally independent of the past given the current state, enabling recursive value decompositions that are the foundation of all dynamic programming and RL algorithms.

1.1 Formal Specification and the Markov Property

A finite MDP is a tuple (S, A, P, R, γ) where S is a finite state space, A a finite action space, $P: S \times A \times S \to [0,1]$ the transition kernel ($P(s' \mid s,a)$ is the probability of transitioning to s' from s under action a), $R: S \times A \to \mathbb{R}$ the reward function, and $\gamma \in [0,1)$ the discount factor. A **policy** $\pi: S \to \Delta(A)$ maps states to distributions over actions. The **Markov property** states that $P(s_{t+1} \mid s_0,a_0,\ldots,s_t,a_t) = P(s_{t+1} \mid s_t,a_t)$, making the transition kernel a sufficient statistic of the history for predicting the future.

1.2 The Bellman Equations

The **state value function** $V^\pi(s) = \mathbb{E}_\pi[\sum_t \gamma^t r_t \mid s_0=s]$ and **action-value function** $Q^\pi(s,a) = \mathbb{E}_\pi[\sum_t \gamma^t r_t \mid s_0=s, a_0=a]$ satisfy the Bellman consistency equations:

$$V^\pi(s) = \sum_a \pi(a \mid s)[R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^\pi(s')]$$

$$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \sum_{a'} \pi(a' \mid s') Q^\pi(s',a')$$

Theorem 1.1 (Bellman Optimality and Contraction)

The optimal value function $V^*(s) = \max_\pi V^\pi(s)$ satisfies the Bellman optimality equation: $V^*(s) = \max_a [R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V^*(s')]$. The Bellman optimality operator $T^*: \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$, $(T^*V)(s) = \max_a [R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V(s')]$ is a γ-contraction in the ℓ^∞ norm: $\|T^*V - T^*U\|_\infty \le \gamma \|V - U\|_\infty$. By Banach's fixed-point theorem, T^* has a unique fixed point V^*, and value iteration $V_{k+1} = T^* V_k$ converges geometrically: $\|V_k - V^*\|_\infty \le \gamma^k \|V_0 - V^*\|_\infty / (1-\gamma)$.

1.3 Q-Learning, TD Learning, and Function Approximation

Q-learning (Watkins and Dayan, 1992) is an off-policy TD algorithm that updates:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)]$$

The DQN (Mnih et al., 2015) extension to deep networks introduces two stabilising mechanisms: **experience replay** (storing transitions in a buffer and sampling random minibatches, breaking correlations between consecutive updates) and **target networks** (maintaining a slowly-updated copy Q_{θ^-} to stabilise the TD target $y = r + \gamma \max_{a'} Q_{\theta^-}(s',a')$). Double DQN addresses overestimation bias by decoupling action selection (online network) from action evaluation (target network).

1.4 Distributional RL and Risk-Sensitive Criteria

Standard RL maximises the expected return. **Distributional RL** (Bellemare et al., 2017) maintains the full **return distribution** $Z^\pi(s,a) \sim \sum_t \gamma^t r_t$ rather than just its mean. The distributional Bellman operator is:

$$Z^\pi(s,a) =^d R(s,a) + \gamma\, Z^\pi(s', \pi(s'))$$

where $=^d$ denotes equality in distribution. C51 (Bellemare et al.) parameterises the return distribution as a categorical distribution over a fixed support. The Wasserstein contraction property $W_p(\mathcal{T}^\pi Z, \mathcal{T}^\pi Z') \le \gamma\, W_p(Z, Z')$ guarantees convergence. Distributional RL leads to improved policies because the uncertainty in the return distribution contains information about the risk of actions that the mean discards — agents that are sensitive to variance (risk-averse) or that exploit variance (risk-seeking) can be derived from the full distribution.

Chapter 2: Policy Gradients

Policy gradient methods optimise the policy directly via gradient ascent on the expected return, without maintaining a value function as the primary object. They are well-suited to continuous action spaces, stochastic policies, and problems with complex constraint structures. Their theoretical foundation is the Policy Gradient Theorem, which provides an unbiased gradient estimator requiring no knowledge of the transition dynamics.

2.1 The Policy Gradient Theorem

Theorem 2.1 (Policy Gradient Theorem, Sutton et al. 2000)

Let $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_t \gamma^t r_t]$ be the expected discounted return. The gradient is:

∇_θ J(θ) = 𝔼_{τ~π_θ}[Σ_t ∇_θ log π_θ(a_t|s_t) · Q^{π_θ}(s_t,a_t)]This holds for any differentiable policy and any MDP regardless of the transition dynamics P — the gradient does not involve ∇_θ P. The quantity ∇_θ log π_θ(a_t|s_t) is the score function (log-derivative) of the policy. The product with Q^π is the reinforce signal: actions with above-average Q are reinforced, below-average actions are suppressed.

The **baseline subtraction** identity states that for any function $b(s)$ depending only on state: $\mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) \cdot b(s)] = 0$. This allows subtracting a state-dependent baseline without introducing bias: $\nabla_\theta J = \mathbb{E}[\nabla_\theta \log \pi_\theta(a_t|s_t) \cdot (Q^\pi(s_t,a_t) - V^\pi(s_t))] = \mathbb{E}[\nabla_\theta \log \pi_\theta(a_t|s_t) \cdot A^\pi(s_t,a_t)]$ where $A^\pi = Q^\pi - V^\pi$ is the **advantage function**. Advantage-based gradients have substantially lower variance than Q-based gradients because A measures relative quality of actions rather than absolute return.

2.2 PPO: Proximal Policy Optimisation

TRPO (Schulman et al., 2015) constrains the policy update to a trust region by solving $\max_\theta \mathbb{E}[r_t(\theta)\hat{A}_t]$ subject to $\mathbb{E}[D_{KL}(\pi_{\theta_{old}} \| \pi_\theta)] \le \delta$ where $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{old}}(a_t|s_t)$ is the importance ratio. PPO (Schulman et al., 2017) replaces the KL constraint with a clipped objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)\hat{A}_t,\ \mathrm{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t)]$$

The clip operation prevents excessively large policy updates: when $\hat{A}_t > 0$, r_t is clipped at $1+\varepsilon$ (no incentive to increase the ratio beyond this); when $\hat{A}_t < 0$, r_t is clipped at $1-\varepsilon$. PPO achieves TRPO-level stability without the conjugate gradient inner loop, making it the dominant policy gradient method in practice.

Chapter 3: Maximum Entropy RL — SAC and TD3

Maximum entropy reinforcement learning augments the standard RL objective with an entropy bonus, incentivising policies to be as stochastic as possible while achieving high reward. This framework has a principled Bayesian interpretation and leads to algorithms with superior exploration and robustness properties compared to deterministic policies.

3.1 The MaxEnt Objective and Soft Bellman Equations

The MaxEnt RL objective is: $J(\pi) = \Sigma_t \mathbb{E}[(r(s_t,a_t) + \alpha H(\pi(\cdot|s_t)))]$ where $H(\pi(\cdot|s)) = -\Sigma_a \pi(a|s) \log \pi(a|s)$ is the policy entropy and α is the temperature parameter. The corresponding soft Bellman equations are:

$$V_{soft}^{\pi}(s) = \mathbb{E}_a[Q_{soft}^{\pi}(s,a) - \alpha \log \pi(a|s)]$$

$$Q_{soft}^{\pi}(s,a) = R(s,a) + \gamma \mathbb{E}_{s'}[V_{soft}^{\pi}(s')]$$

Theorem 3.1 (Optimal MaxEnt Policy)

The policy maximising the MaxEnt objective with temperature α is: $\pi^*(a|s) \propto \exp(Q_{soft}^*(s,a)/\alpha)$. This is a Boltzmann (Gibbs) distribution over actions, with temperature controlling the tradeoff between exploitation (low $\alpha \to$ concentrated on high-Q actions) and exploration (high $\alpha \to$ uniform distribution). The optimal policy satisfies a soft policy improvement theorem: for any policy π, the policy $\tilde{\pi}(a|s) \propto \exp(Q^{\pi}(s,a)/\alpha)$ achieves $V_{soft}^{\tilde{\pi}}(s) \geq V_{soft}^{\pi}(s)$ everywhere, guaranteeing monotone improvement.

3.2 Soft Actor-Critic (SAC)

SAC (Haarnoja et al., 2018) implements MaxEnt RL for continuous actions with a squashed Gaussian policy: $a_t = \tanh(\mu_\theta(s_t) + \sigma_\theta(s_t) \odot \varepsilon)$, $\varepsilon \sim \mathcal{N}(0,I)$. The critic learns two Q-functions Q_{w_1}, Q_{w_2} and uses the minimum to mitigate overestimation. The actor maximises:

$$J_\pi(\theta) = \mathbb{E}_{s\sim D,\ \varepsilon\sim\mathcal{N}}[\alpha \log \pi_\theta(f_\theta(\varepsilon;s)|s) - \min(Q_{w_1}(s,f_\theta(\varepsilon;s)), Q_{w_2}(s,f_\theta(\varepsilon;s)))]$$

The temperature α can be tuned automatically by minimising $J(\alpha) = \mathbb{E}[-\alpha \log \pi_\theta(a_t|s_t) - \alpha \bar{H}]$ where $\bar{H}$ is a target entropy (e.g., $-\dim(A)$). SAC achieves state-of-the-art sample efficiency on continuous control benchmarks and remarkable robustness to hyperparameter choices.

3.3 TD3: Addressing Q-value Overestimation

TD3 (Fujimoto et al., 2018) is a deterministic actor-critic addressing three failure modes of DDPG: overestimation via **clipped double Q-learning** (using the minimum of two critics for the target); high variance via **target policy**

smoothing (adding clipped noise to target actions: $\tilde{a} = clip(\mu_{\theta'} + clip(\varepsilon, -c, c), a_{low}, a_{high})$); and instability via **delayed policy updates** (updating the actor every d=2 critic updates). The combination produces sample-efficient, stable learning for continuous control tasks.

Chapter 4: Model-Based RL — World Models and Latent Planning

Model-based RL (MBRL) augments the policy with a learned model of the environment dynamics, enabling planning in imagination. The fundamental question is when and how to trust the model: a perfect model allows arbitrary planning, but model errors compound, potentially leading to catastrophic policy exploitation of model inaccuracies.

4.1 The MBPO Error Bound and Short Rollout Principle

Theorem 4.1 (MBPO Branched Rollout Bound, Janner et al. 2019)

Let $\varepsilon_m = \max_t \mathbb{E}_{s\sim\pi}[D_{TV}(p(s'|s,a) \| \hat{p}(s'|s,a))]$ be the model error and $\varepsilon_\pi = \max_t \mathbb{E}_{s\sim\hat{p}}[D_{TV}(\pi(a|s) \| \pi_D(a|s))]$ the policy error. For k-step model rollouts branching from real data, the return difference between the model-based and true policy is bounded by: $|\eta[\pi] - \eta_{model}[\pi]| \le 2\gamma(\gamma^k \varepsilon_m + (1-\gamma^k)(\varepsilon_m + \varepsilon_\pi)) / (1-\gamma)^2$. The bound is minimised at a short rollout length $k = O(\log(\varepsilon_m)/\log(\gamma))$, independent of π complexity. This justifies MBPO's key design: short (1–5 step) model rollouts that generate synthetic transitions for off-policy RL, gaining sample efficiency without catastrophic model exploitation.

4.2 Dreamer and Latent Space Planning

Dreamer (Hafner et al., 2020) learns a world model with a **Recurrent State Space Model (RSSM)** that factors the latent state into deterministic h and stochastic z components: $h_t = f(h_{t-1}, z_{t-1}, a_{t-1})$ and $z_t \sim q_\varphi(z_t|h_t, o_t)$. Policies and value functions are trained entirely from imagined trajectories within the latent model, never rolling out in the real environment. The model is trained by maximising the ELBO of the observation sequence:

$$\mathcal{L}_{model} = \mathbb{E}_q[\Sigma_t \log p(o_t|h_t,z_t) - \beta\, D_{KL}(q(z_t|h_t,o_t) \| p(z_t|h_t))]$$

DreamerV3 introduces **symlog predictions** (symmetric log transform $f(x) = sign(x)\cdot ln(|x|+1)$ applied to targets) and **free bits** (D_{KL} is clipped from below at a free-bits threshold), enabling training with a single set of universal hyperparameters across tasks spanning orders of magnitude in reward scale.

Chapter 5: RLHF, DPO, and the Mathematics of Alignment

The alignment problem asks how to ensure that learned systems pursue objectives that are beneficial to humans. In the era of large language models, this requires training systems on human preferences — a challenging signal that is expensive to collect, subject to inconsistency, and potentially gameable. The mathematical framework of RLHF and its successors provides the formal tools for this problem.

5.1 The Bradley-Terry Model and Reward Learning

RLHF assumes that human preferences between response pairs (y_w, y_l) given prompt x follow the Bradley-Terry model:

$$p^*(y_w \succ y_l \mid x) = \sigma(r^*(x, y_w) - r^*(x, y_l))$$

where $r^*(x,y)$ is the latent reward function and σ is the logistic function. A reward model $r_\varphi(x,y)$ is trained by maximum likelihood on a preference dataset $\mathcal{D} = \{(x, y_w, y_l)\}$:

$$\mathcal{L}_{RM}(\varphi) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}[\log \sigma(r_\varphi(x,y_w) - r_\varphi(x,y_l))]$$

The trained reward model is then used to fine-tune the language model policy via PPO, maximising $\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r_\varphi(x,y)] - \beta\, D_{KL}(\pi_\theta \| \pi_{ref})$ where π_{ref} is the reference (SFT) policy and β controls the divergence penalty.

Theorem 5.1 (Optimal RLHF Policy)

The optimal policy for the KL-constrained RLHF objective max_π 𝔼[r(x,y)] − β D_KL(π ‖ π_ref) has the closed-form solution: π*(y|x) = π_ref(y|x) exp(r(x,y)/β) / Z(x)where Z(x) = Σ_y π_ref(y|x) exp(r(x,y)/β) is the partition function. This is a Boltzmann distribution over responses, shifted from the reference policy by the reward signal with temperature β. The partition function Z(x) is intractable to compute exactly — this is the fundamental reason RLHF

requires PPO or DPO: neither can directly sample from π^* without approximation.

5.2 Direct Preference Optimisation (DPO)

DPO (Rafailov et al., 2023) eliminates the reward model by reparameterising $r^*(x,y)$ in terms of the optimal policy π^* and reference policy π_{ref} via the closed-form expression in Theorem 5.1. Substituting $r(x,y) = \beta \log(\pi^*(y|x)/\pi_{ref}(y|x)) + \beta \log Z(x)$ into the Bradley-Terry model and observing that $\log Z(x)$ cancels in the preference difference yields:

$$\mathcal{L}_{DPO}(\theta) = -\mathbb{E}[\log \sigma(\beta \log(\pi_\theta(y_w|x)/\pi_{ref}(y_w|x)) - \beta \log(\pi_\theta(y_l|x)/\pi_{ref}(y_l|x)))]$$

DPO is a binary cross-entropy objective over implicit reward differences, requiring no explicit reward model, no RL optimisation loop, and no sampling during training. The implicit reward of DPO is $\hat{r}_\theta(x,y) = \beta \log(\pi_\theta(y|x)/\pi_{ref}(y|x))$ — the log-ratio of the current to reference policy, scaled by temperature. Analysis shows that DPO optimises the same objective as RLHF+PPO under the Bradley-Terry model, but the finite-sample behaviour differs due to the language model's inability to place probability mass on all possible responses.

5.3 KTO, SimPO, and Identity-Mapped Preference Optimisation

KTO (Ethayarajh et al., 2024) replaces the pairwise preference model with a **prospect-theoretic** framework based on Kahneman-Tversky utility: humans weight gains and losses asymmetrically, with losses felt more acutely. The KTO objective uses unpaired feedback (individual good/bad examples rather than preference pairs):

$$\mathcal{L}_{KTO}(\theta) = \mathbb{E}[(\lambda_D - \sigma(\hat{r}(x,y) - z_0))\cdot 1[y \text{ desirable}] + (\lambda_U - \sigma(z_0 - \hat{r}(x,y)))\cdot 1[y \text{ undesirable}]]$$

where $z_0 = \mathbb{E}_{x' \sim \mathcal{D}}[D_{KL}(\pi_\theta(y|x') \| \pi_{ref}(y|x'))]$ is the KL reference point. KTO requires only binary quality labels rather than preference comparisons, reducing annotation cost substantially. **SimPO** (Meng et al., 2024) modifies DPO by using length-normalised rewards $r(x,y) = (\beta/|y|) \log \pi_\theta(y|x)$ and a target margin γ, eliminating the reference model entirely.

5.4 Constitutional AI and Scalable Oversight

Constitutional AI (Bai et al., 2022) addresses the scalability of human oversight: as AI systems become more capable, humans cannot reliably evaluate all outputs. The approach has two phases: **supervised learning from AI feedback** (the model critiques and revises its own outputs according to a constitution of principles) and **RLHF from AI feedback** (RLAIF, where an AI classifier replaces human annotators for preference labelling). The mathematical structure is a fixed-point argument: the model improves by iterating the critique-revision loop until it satisfies the constitutional principles in expectation.

Chapter 6: Emergent Properties and In-Context Learning

Large language models exhibit capabilities that emerge abruptly with scale: behaviours absent at smaller scales that appear suddenly above a threshold. Understanding these phenomena requires mathematical tools that span information theory, learning theory, and the theory of computation.

6.1 In-Context Learning as Implicit Bayesian Inference

ICL presents a model with k examples $(x_1,y_1),\ldots,(x_k,y_k)$ and a query x_{k+1} and predicts y_{k+1} without parameter updates. Xie et al. (2022) model ICL as Bayesian inference over a latent concept C: the pretraining distribution is a mixture $p(x) = \Sigma_C\, p(C)p(x \mid C)$, and the language model implicitly maintains a posterior $p(C \mid x_1,\ldots,x_k)$ which concentrates on the true concept as more examples arrive. The ICL prediction is then $p_{LM}(y \mid context) \approx \mathbb{E}_{C\sim p(C \mid context)}[p(y \mid x_{k+1},C)]$.

Akyürek et al. (2023) provide an alternative mechanistic interpretation: transformer layers can implement gradient descent steps, so ICL corresponds to running ridge regression in the forward pass. Specifically, for a one-hidden-layer transformer with specific weight initialisation, the attention update $W_Q^T W_K$ implements a gradient descent step on the in-context examples, with the value matrix W_V learning the mapping from features to labels.

6.2 Chain-of-Thought and Computational Depth

Standard transformers with constant depth and width are limited to the circuit complexity class TC^0 (constant-depth threshold circuits). Feng et al. (2023) show that chain-of-thought reasoning, which increases the effective depth by generating intermediate tokens, allows the network to compute functions outside TC^0 — specifically up to NC^1 (log-depth circuits). This provides a theoretical justification for CoT beyond empirical observations: the intermediate tokens function as a scratchpad that increases computational depth, enabling tasks like counting, sorting, and arithmetic that require depth super-constant in input length.

Chapter 7: Mechanistic Interpretability

Mechanistic interpretability seeks to reverse-engineer the internal computations of neural networks: to identify, at the level of weights and activations, what algorithms a network has learned. This goes beyond the "what does this neuron respond to" question of classical feature visualisation, toward the "what circuit implements this capability" question that yields a complete computational account.

7.1 Circuits and the Modularity Hypothesis

The **circuits hypothesis** (Olah et al., 2020) proposes that neural networks implement identifiable, modular computational circuits: small subgraphs of the computational graph that implement a recognisable function, composed from universal feature families (curve detectors, frequency detectors, multimodal neurons). A **circuit** is formally a subgraph (N, E) of the full computation graph $G = (V, E)$ sufficient to explain a specific behaviour, in the sense that ablating complement nodes does not alter the behaviour on the relevant distribution.

7.2 Superposition and the Geometry of Features

The **superposition hypothesis** (Elhage et al., 2022) proposes that neural networks represent more features than they have dimensions by encoding features as near-orthogonal directions in activation space: if there are F features but only d dimensions with $F \gg d$, the network can store $O(F)$ features using interference at rate $O(F/d)$ per feature. The optimal solution for storing F features of importance I_f in dimension d allocates dimensions to features in

proportion to I_f, and packs low-importance features as antipodal pairs on the same dimension.

Theorem 7.1 (Superposition Capacity, Elhage et al. 2022)

Consider a model that stores features $\{f_i\}$ as directions $\{u_i\}$ in $\mathbb{R}^d$, reconstructing each as $f_i \approx u_i^T h$ where h is the hidden activation. For sparse features (each present with probability $p \ll 1$), the total interference loss is $\mathbb{E}[\text{loss}] = \Sigma_i (1 - \|u_i\|^2)^2 I_i + \Sigma_{i \neq j} (u_i \cdot u_j)^2 I_i I_j \cdot p$. When $d < F$, the minimum is achieved by the polytope code: features placed at vertices of a regular polytope in $\mathbb{R}^d$. For $p \to 0$, interference goes to zero and $F/d \to \infty$, enabling unbounded superposition in the limit of perfect sparsity.

7.3 Sparse Autoencoders for Feature Discovery

Sparse autoencoders (Cunningham et al., 2023) learn a dictionary of $M \gg d$ feature directions by training a single-layer autoencoder with L_1 regularisation on the activations h of a language model:

$$\mathscr{L}_{SAE} = \|h - D \cdot ReLU(Eh + b)\|^2 + \lambda \|ReLU(Eh + b)\|_1$$

where $E \in \mathbb{R}^{M \times d}$ is the encoder and $D \in \mathbb{R}^{d \times M}$ the decoder (with unit-norm columns to prevent trivial solutions). The L_1 penalty induces sparsity in the feature activations, decomposing the polysemantic, dense activation vector h into a sparse sum of monosemantic feature vectors. Empirically, the features discovered by SAEs are human-interpretable (representing named entities, syntactic structures, emotional tones) at a granularity not achievable by examining individual neurons.

7.4 Induction Heads and the Mechanistic Basis of ICL

Induction heads (Olsson et al., 2022) are a specific two-head circuit in transformer language models that implement in-context copying. Given a sequence $\ldots A \ldots AB$, an induction head predicts B by: (1) a **previous token head** in layer 0 that attends to the token immediately preceding the current position, writing the previous token into the residual stream; (2) an **induction head** in layer 1 that uses Q-K matching to find where the current token appeared earlier in context, attending to the position just after that match. This two-layer circuit is sufficient to implement arbitrary in-context sequence completion and represents the mechanistic basis for ICL identified in Theorem 7.1.

Concluding Remarks on Part V

Reinforcement learning and alignment present perhaps the deepest mathematical challenges in machine learning: the credit assignment problem, the exploration-exploitation tradeoff, the instability of off-policy learning with function approximation, and the misspecification problem in reward learning. The tools developed here — from Bellman contraction theorems to the DPO reparameterisation — provide rigorous foundations for systems that must act wisely in uncertain environments and in accordance with human values. The mechanistic interpretability programme closes the circle by providing the mathematical tools to verify, post-hoc, that learned systems have implemented safe and interpretable algorithms.

PART VI

ADVANCED MATHEMATICAL STRUCTURES

Part VI of the Mathematics of Deep Learning

Statistical Physics · Algebraic Topology · Causal Inference · Scaling Laws · Fine-Tuning · Multimodal Models

Introduction to Part VI

The final part of this treatise assembles the advanced mathematical frameworks that complete the intellectual architecture of deep learning theory. Each chapter draws on a distinct mathematical tradition — statistical physics, algebraic topology, causal inference, combinatorics, distributed systems, and optimal transport — applying it to foundational questions about why neural networks generalise, what the limits of scale are, how to efficiently adapt large pre-trained models, and how to align representations across modalities. The common thread is that empirical phenomena in deep learning are not mysterious: they are consequences of precise mathematical structures that rigorous analysis can identify and characterise.

Chapter 1: Statistical Physics of Learning — Spin Glasses, Replicas, and Free Probability

The connection between neural networks and statistical physics is not merely analogical. Many phenomena in deep learning — phase transitions, the shape of loss landscapes, the relationship between network size and generalisation — have precise counterparts in the theory of disordered systems that statistical mechanics has developed since the 1970s.

1.1 The Modern Hopfield Network and Exponential Memory

Classical Hopfield networks (Hopfield, 1982) store M patterns $\{\xi^\mu\}$ in a weight matrix $W = (1/n)\Sigma^\mu\ \xi^\mu(\xi^\mu)^T$ with energy $E(x) = -\frac{1}{2}x^TWx$. The Amit-Gutfreund-Sompolinsky (AGS) analysis via the replica method yields a critical capacity $\alpha c = M/n \approx 0.138$: below this threshold, patterns are stable attractors with high probability; above it, the network undergoes a first-order phase transition to a spin glass phase where retrieval fails catastrophically.

Theorem 1.1 (AGS Capacity Theorem via Replica Method)

For a Hopfield network with n neurons and $M = \alpha n$ random ±1 patterns, the replica symmetric free energy per neuron is: $f = -\frac{1}{2}(1-\alpha/\alpha c)$ + contributions from saddle point equationsThe critical load $\alpha c \approx 0.138$ is determined by the condition that the replica-symmetric solution becomes unstable (Almeida-Thouless instability). Below αc, patterns are retrieved with average overlap $m = \mathbb{E}[\xi^\mu \cdot x]/n \to 1$ from any initial condition within Hamming distance $0.1n$ of the pattern. Above αc, the

system falls into a spin glass phase with exponentially many metastable states of low energy, none of which correspond to stored patterns.

1.2 The Replica Method and Free Energy Calculations

The partition function $Z = \Sigma_x \exp(-\beta E(x))$ of a disordered system (e.g., a Hopfield network with random patterns) cannot be computed directly. The replica trick computes the quenched free energy $F = -(1/\beta)\mathbb{E}_{\xi}[\log Z]$ via the identity $\log Z = \lim_{n\to 0} (Z^n - 1)/n$. For integer n, $\mathbb{E}[Z^n]$ is the partition function of n replicas of the system with the same disorder ξ, hence tractable. Analytic continuation to $n \to 0$ yields the quenched average.

The **replica symmetric (RS) ansatz** assumes that the overlap matrix between replicas, $q_{ab} = (1/n)\Sigma_i x_i^a x_i^b$, has the form $q_{ab} = q$ for all $a \neq b$ — all replicas are statistically equivalent. RS breaks down in the spin glass phase, requiring **Parisi's replica symmetry breaking (RSB)** scheme, where the overlap matrix is a hierarchical structure parameterised by a function $q(x)$ on $[0,1]$. This structure predicts ultrametricity of the energy landscape: states are organised in a hierarchical tree, with distances satisfying the ultrametric inequality $d(x,z) \leq \max(d(x,y), d(y,z))$.

1.3 Free Probability and the Marchenko-Pastur Law Revisited

Free probability (Voiculescu, 1991) is the non-commutative analogue of classical probability theory, in which independence of random variables is replaced by **freeness** of random matrices. For large random matrices, free probability provides exact calculations of eigenvalue distributions (spectral measures) that converge as matrix dimension tends to infinity.

The **R-transform** $R_A(z)$ of a random matrix A is the analogue of the cumulant generating function. For free random matrices A and B: $R_{A+B}(z) = R_A(z) + R_B(z)$. This free additive convolution theorem enables exact computation of the spectral density of sums of random matrices — directly relevant to the spectrum of neural network weight updates: if the weight update ΔW is "free" from the pre-trained weight W_0 (a condition approximately satisfied when the update is in a random direction), then the spectral density of $W = W_0 + \Delta W$ is the free convolution of the spectra of W_0 and ΔW.

1.4 The Jamming Transition and Loss Landscape Geometry

The **jamming transition** in physics (the transition from a flowing to a rigid amorphous solid as density increases) has a precise analogue in neural network training. An overparameterised network with $P > N$ parameters has more degrees of freedom than data constraints, placing it in the **jamming-above** phase where the loss landscape has no barriers to achieving zero training loss. An underparameterised network is in the jamming-below phase where perfect fit is geometrically impossible.

Sagun et al. (2017) show that the Hessian spectrum of overparameterised networks has a large bulk near zero (bulk eigenvalues corresponding to flat directions) plus a small number of outlier eigenvalues (corresponding to informative curvature). The bulk has magnitude $O(N/P)$, vanishing as overparameterisation increases. This explains empirically why overparameterised networks trained with SGD find flat minima with good generalisation: the noise of SGD effectively selects among the continuum of global minima, preferring those in wide flat regions (large ε-ball with small loss) that correspond to good generalisation by PAC-Bayes bounds.

Chapter 2: Algebraic Topology Applied to Machine Learning

Topology studies properties of spaces that are preserved under continuous deformations: connectedness, holes, and higher-dimensional voids. Applied to machine learning, topological methods provide invariants of data manifolds and function landscapes that are robust to noise and provide structural information inaccessible to geometric or statistical methods.

2.1 Persistent Homology

Given a finite point cloud $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, the **Čech complex** $\check{C}_\varepsilon$ at scale ε has the same homotopy type as the union of ε-balls around the points. As ε increases, topological features (connected components, loops, voids) are born and die. **Persistent homology** tracks this birth-death history.

Theorem 2.1 (Persistence Structure Theorem)

The k-th persistent homology of a filtration $\{X\varepsilon\}_{\varepsilon\geq 0}$ decomposes uniquely (over a field) as a direct sum of interval modules $[b_i, d_i)$: $H_k(\{X\varepsilon\}) \cong \oplus_i \mathbb{k}[b_i, d_i)$. Each interval $[b_i, d_i)$ records the birth ($\varepsilon = b_i$) and death ($\varepsilon = d_i$) of a topological feature of dimension k (0: connected components, 1: loops, 2: voids). The persistence diagram $\{(b_i, d_i)\}$ is a multiset in $\mathbb{R}^2$ encoding all topological features. Features near the diagonal ($d_i - b_i \approx 0$) are topological noise; features far from the diagonal are genuine topological structure.

2.2 Sheaf Theory and the Connection Laplacian

A **cellular sheaf** $\mathscr{F}$ on a graph $G = (V, E)$ assigns a vector space $\mathscr{F}(v)$ to each vertex and $\mathscr{F}(e)$ to each edge, together with linear restriction maps $\mathscr{F}_{v \trianglelefteq e}: \mathscr{F}(v) \to \mathscr{F}(e)$. The **sheaf Laplacian** $\Delta_{\mathscr{F}} = B^T B$ (where B is the coboundary operator of the sheaf) generalises the graph Laplacian to signals that are locally consistent only in the sheaf-theoretic sense.

The **global sections** $H^0(G, \mathscr{F}) = \ker(\Delta_{\mathscr{F}})$ are the globally consistent assignments — agreement configurations across the entire graph. For opinion dynamics, the sheaf assigns opinion vectors to nodes and $\mathscr{F}_{v \trianglelefteq e}$ encodes how node v transforms its opinion to compare with the e-direction. The sheaf Laplacian dynamics $\dot{x} = -\Delta_{\mathscr{F}} x$ model opinion evolution with nontrivial agreement structures, recovering classic opinion dynamics models as special cases while capturing directional and asymmetric influence.

2.3 Grokking: Delayed Generalisation as a Topological Phase Transition

Grokking (Power et al., 2022) refers to the phenomenon where a network first memorises the training set (near-zero training loss, near-chance validation loss) and then, after extended training, generalises (validation loss drops to near-zero), with the transition separated by tens of thousands of gradient steps. Analysis via persistent homology and weight norm dynamics reveals that grokking corresponds to a **phase transition** in the representation geometry: the representation first organises into a structure that separates training examples (memorisation) and then reorganises into a lower-complexity, more symmetric structure that supports generalisation.

Nanda et al. (2023) provide a mechanistic account for modular addition: the network learns a **Fourier representation** of the input, implementing the algorithm $a + b \bmod p$ via circular embeddings $(\cos(2\pi ka/p), \sin(2\pi ka/p))$ composed with a product-to-sum trigonometric identity. The grokking transition corresponds to the network discovering this Fourier structure, which requires less weight norm to represent than the memorisation solution — explaining why the transition occurs under weight decay: the L_2 regularisation makes the Fourier solution thermodynamically preferred.

Chapter 3: Causal Inference and Out-of-Distribution Generalisation

Statistical learning theory provides bounds on in-distribution generalisation: given sufficient data from a distribution P, a learned model will perform well on new samples from P. But deployed models face distributional shift: the test distribution Q differs from training P. Causal inference provides the mathematical framework for identifying which aspects of a distribution are stable across environments and which are spurious correlations that will fail under shift.

3.1 Structural Causal Models and the Do-Calculus

A **Structural Causal Model (SCM)** consists of a DAG G over variables $X_1, \ldots, X_n$ and structural equations $X_i = f_i(PA(X_i), U_i)$ where $PA(X_i)$ are the parents of X_i in G and U_i are independent exogenous noise variables. The **do-operator** $do(X_i = x)$ replaces the structural equation for X_i with the constant x, cutting all incoming edges to X_i in G. The interventional distribution $p(X_j \mid do(X_i = x))$ answers the causal question: what would X_j be if we set $X_i = x$?

Theorem 3.1 (Pearl's Do-Calculus Completeness)

The do-calculus (three rules applied to the causal graph G and its c-components) is complete for identifying causal effects: a causal effect $P(Y \mid do(X))$ is identified from observational data if and only if it can be derived using the do-calculus rules. Non-identifiability occurs when the graph has unobserved confounders (hidden common causes) that create backdoor paths from X to Y that cannot be blocked by conditioning on observed variables. The front-door criterion and instrumental

variables represent two families of identification strategies for specific graph structures.

3.2 Invariant Risk Minimisation

IRM (Arjovsky et al., 2019) seeks a feature representation Φ such that the optimal linear classifier on top of Φ is the same across all training environments $\{e \in \mathcal{E}_{train}\}$:

$$\min_{\Phi, w} \sum_e R^e(w \circ \Phi) \quad \text{s.t.} \quad w \in \operatorname{argmin}_{w'} R^e(w' \circ \Phi) \;\; \forall e \in \mathcal{E}_{train}$$

The invariance constraint expresses the causal hypothesis that the features used by Φ are causally related to the label Y (and hence stable across environments) rather than merely correlated in the training data. The practical relaxation IRM-v1 adds a gradient penalty: $\|\nabla_{w|w=1} R^e(w \circ \Phi)\|^2 \to 0$ (the optimal dummy classifier on top of Φ in each environment should be the all-ones weight vector, meaning Φ already carries all information needed).

3.3 Distributionally Robust Optimisation

DRO (Sagawa et al., 2020) minimises the **worst-group risk**: $\min_\theta \max_{g \in G} R^g(\theta)$ where G is a collection of groups (e.g., demographic subgroups or environments). The Lagrangian formulation with group weights q_g gives:

$$\min_\theta \max_{q \in \Delta(G)} \sum_g q_g R^g(\theta) = \min_\theta \max_{P' \in \mathcal{P}} \mathbb{E}_{P'}[\ell(f_\theta(x), y)]$$

where $\mathcal{P}$ is the uncertainty set (convex hull of group distributions). DRO with χ^2 uncertainty sets ($D_{\chi^2}(P' \| P) \le \varepsilon$) is particularly tractable: the dual problem has a closed form involving the conditional variance of the loss, and the empirical optimum can be computed by upweighting examples with high loss. The key connection to IRM is that both methods seek representations robust to distributional shift, but IRM exploits causal structure while DRO provides adversarial robustness to any shift within the uncertainty set.

Chapter 4: Scaling Laws and Emergent Capabilities

4.1 The Kaplan and Chinchilla Scaling Laws

Kaplan et al. (2020) empirically established that language model loss scales as a power law in three quantities: model parameters N, dataset tokens D, and compute $C \approx 6ND$:

$$L(N) \approx (N_c/N)^{\alpha_N}, \quad L(D) \approx (D_c/D)^{\alpha_D}, \quad \alpha_N \approx 0.076, \alpha_D \approx 0.095$$

The compute-optimal frontier — the Pareto-optimal tradeoff between N and D for fixed C — was derived by Hoffmann et al. (2022) (Chinchilla): minimising $L(N,D) = E + A/N^{\alpha} + B/D^{\beta}$ subject to $C = 6ND$ gives the optimal allocation $N^* \propto C^{0.5}$, $D^* \propto C^{0.5}$, implying equal scaling of parameters and tokens. This overturned the Kaplan recommendation (scale N faster than D) and motivates the Chinchilla result: a 70B model trained on 1.4T tokens outperforms a 280B model trained on 300B tokens at the same compute budget.

Theorem 4.1 (Optimal Compute Allocation)

Under the loss decomposition $L(N,D) = E + A \cdot N^{-\alpha} + B \cdot D^{-\beta}$ and compute constraint $C = 6ND$, the compute-optimal $(N^*(C), D^*(C))$ satisfies: $N^*(C) = (\alpha A/\beta B)^{1/(\alpha+\beta)} \cdot C^{\beta/(\alpha+\beta)}$ $D^*(C) = (\beta B/\alpha A)^{1/(\alpha+\beta)} \cdot C^{\alpha/(\alpha+\beta)}$For the Chinchilla estimates $\alpha \approx \beta \approx 0.5$, this gives $N^* \propto D^* \propto \sqrt{C}$, recovering equal scaling. The key implication is that for fixed C, there is a unique optimal model size — both larger and smaller models achieve higher loss at the same compute cost. This is the fundamental justification for the Chinchilla scaling law.

4.2 Emergent Capabilities as Phase Transitions

Wei et al. (2022) define emergent capabilities as abilities that are "absent in smaller models and present in larger models" with a discontinuous transition — the capability appears to jump from near-zero to significant performance. Schaeffer et al. (2023) provide a metric-dependent explanation: many apparently discontinuous transitions are artifacts of nonlinear metrics (accuracy, multiple-choice accuracy) applied to smooth underlying improvements. Under linear metrics (continuous-valued, proportional to partial credit), the same models show smooth, gradual improvement.

The rigorous question is whether genuine phase transitions exist — discontinuities in a smooth metric as $N \to \infty$ at fixed C/N. Theoretical results suggest they do for specific tasks: tasks requiring k conditions to be simultaneously satisfied show a threshold effect at the parameter count

needed to solve each condition, with the joint probability of success transitioning from $O(\varepsilon^k)$ to $O(1)$ as each condition is crossed — a binomial composition yielding sharp transitions even in smooth metrics.

Chapter 5: Large-Scale Fine-Tuning

5.1 LoRA: Mathematical Foundations and Rank Selection

LoRA (Hu et al., 2022) parameterises weight updates as $\Delta W = BA$ with $B \in \mathbb{R}^{m \times r}$ and $A \in \mathbb{R}^{r \times n}$, $r \ll \min(m,n)$. As established in Part I (Theorem 1.5), the optimal LoRA approximation achieves Frobenius norm error $\|\Sigma_r\|_F = \sqrt{(\sigma_{r+1}^2 + \ldots + \sigma_R^2)}$ where σ_i are the singular values of the optimal full update ΔW^*. The practical rank selection question — how large should r be? — is answered by examining the spectral decay of ΔW^* on validation data; empirically, $r \in \{4, 8, 16, 32\}$ captures >95% of the Frobenius norm for most NLP tasks.

QLoRA (Dettmers et al., 2023) extends LoRA to 4-bit quantisation: the frozen weights W_0 are stored in NF4 (NormalFloat 4-bit), a data type that minimises quantisation error for normally distributed weights. The key insight is that the quantisation error $\|W_0 - Q(W_0)\|$ does not accumulate in the LoRA path — only BA are updated in full precision, and the LoRA update corrects for quantisation error during training. This allows fine-tuning of 65B parameter models on a single 48GB GPU with <1% performance degradation versus full-precision fine-tuning.

5.2 ZeRO and Fully Sharded Data Parallelism

Data parallel training (DDP) replicates the full model on each GPU and averages gradients across replicas. For a 100B model in mixed precision, this requires $100B \times 16$ bytes ≈ 1.6 TB per GPU — far exceeding GPU memory. ZeRO (Rajbhandari et al., 2020) eliminates this redundancy by sharding the three memory-intensive quantities across N GPUs:

ZeRO-1: Shard optimiser states (Adam stores gradient and second moment = 12 bytes/param). Memory reduction: 4×.

ZeRO-2: Also shard gradients. Memory reduction: 8×.

ZeRO-3: Also shard parameters. Memory reduction: N× (linear in GPU count).

Theorem 5.1 (ZeRO-3 Memory Complexity)

For a model with Ψ parameters trained with N GPUs and Adam optimizer (12 bytes/param for optimizer states, 2 bytes/param fp16 for gradients, 2 bytes/param fp16 for parameters), ZeRO-3 reduces per-GPU memory from 16Ψ bytes (DDP) to (16Ψ/N + communication buffer) bytes. For N=64 GPUs: 64-fold reduction in model-state memory. The communication cost is ∼3Ψ bytes per training step (one allgather for parameters + one reduce-scatter for gradients), comparable to DDP's allreduce cost of 2Ψ bytes. ZeRO-3 achieves near-linear memory scaling with GPU count with only ∼50% communication overhead over DDP.

5.3 Gradient Checkpointing and the Memory-Compute Tradeoff

Full backpropagation requires storing all intermediate activations for use in the backward pass, requiring $O(L \cdot B \cdot d)$ memory for L layers, batch size B, and hidden dimension d. Gradient checkpointing (Chen et al., 2016) saves memory by recomputing activations during the backward pass from periodically saved checkpoints.

Theorem 5.2 (Checkpointing Memory-Compute Tradeoff)

With checkpointing at every √L layers (√L checkpoint layers, each storing activations at cost O(B·d)), total memory is O(√L · B · d) — down from O(L·B·d). The compute cost increases by a factor of ∼1 + 1/√L ≈ 1 (one additional partial forward pass). This √L tradeoff is optimal: checkpointing at every k layers costs O(k · B · d + L/k · B · d) memory, minimised at k = √L giving O(√L · B · d). For L=100 layers, this is a 10× memory reduction at <1% compute overhead.

Chapter 6: Multimodal Models — Geometry of Joint Representation Spaces

Multimodal learning seeks to align representations from different modalities — images, text, audio, video, protein sequences — in a shared embedding space where proximity reflects semantic similarity. The mathematical

challenge is that different modalities have different statistics, different natural geometries, and different information-theoretic relationships to the shared semantic content.

6.1 CLIP and Contrastive Cross-Modal Alignment

CLIP (Radford et al., 2021) trains an image encoder $f: X \to \mathbb{R}^d$ and text encoder $g: T \to \mathbb{R}^d$ jointly on N matched pairs $\{(x_i, t_i)\}_i$ using a symmetric InfoNCE loss over an $N \times N$ similarity matrix:

$$\mathscr{L}_{CLIP} = -(1/2N)\, \Sigma_i \left[\log \exp(f(x_i)\cdot g(t_i)/\tau) / \Sigma_j \exp(f(x_i)\cdot g(t_j)/\tau) \;+\; \log \exp(f(x_i)\cdot g(t_i)/\tau) / \Sigma_j \exp(f(x_j)\cdot g(t_i)/\tau)\right]$$

The first term aligns each image with its text (image-to-text direction); the second aligns each text with its image. At convergence, the optimal encoder satisfies $f(x)\cdot g(t) \propto \log p(x,t)/p(x)p(t)$ — the representations encode the pointwise mutual information between matched image-text pairs. The normalised temperature τ controls the sharpness of the contrastive distribution: small τ makes the loss focus on the hardest negatives, while large τ makes the loss more uniform.

6.2 Optimal Transport for Cross-Modal Alignment

The CLIP loss treats each image-text pair independently, ignoring the geometric structure of the embedding space. **Optimal transport** provides a richer notion of alignment between distributions. The **Wasserstein distance** between the image and text embedding distributions:

$$W_p(\mu_f, \mu_g) = \left(\inf_{\gamma \in \Pi(\mu_f,\mu_g)} \int \|z_f - z_g\|^p \, d\gamma(z_f,z_g)\right)^{1/p}$$

measures the geometric discrepancy between the two embedding distributions. The Sinkhorn algorithm (Cuturi, 2013) computes the regularised OT distance $OT_\varepsilon(\mu_f,\mu_g) = \min_{\gamma \in \Pi} \langle C,\gamma \rangle + \varepsilon\, H(\gamma)$ efficiently via matrix scaling. In practice, OT-based alignment losses produce representations where the **distributions** of embeddings are aligned (not just individual pairs), improving zero-shot transfer to tasks that require reasoning about distributional properties of the representations rather than individual similarities.

6.3 Vision-Language Models and Cross-Attention Conditioning

Flamingo (Alayrac et al., 2022) interleaves a frozen vision encoder (NFNet) with a frozen language model, connected by **Perceiver Resampler** (cross-attention from fixed query tokens to visual features) and **cross-attention layers** inserted into the LM. The Perceiver Resampler compresses variable-length visual features $(T_{vis} \times d_{vis})$ to a fixed number of visual tokens $(N_{vis} \times d_{lm})$:

$$z = \text{CrossAttn}(Q, K{=}V{=}\text{visual features}),\ Q \in \mathbb{R}^{N_{vis} \times d_{lm}}$$

The cross-attention layers in the LM implement the conditioning: language tokens attend to visual tokens at every layer, enabling visual information to modulate the language generation at every level of the LM hierarchy. Flamingo's training on interleaved image-text sequences enables few-shot visual question answering without task-specific fine-tuning — a major advance over encoder-only vision-language models like CLIP.

6.4 Unified Multimodal Architectures and Tokenisation

The trend toward **unified multimodal architectures** (LLaVA, Gemini, GPT-4V) treats all modalities as token sequences and uses a single autoregressive transformer. Images are tokenised via VQ-VAE discrete codes (enabling generation) or CLIP patch features (enabling understanding); audio via EnCodec tokens; video via spatiotemporal patches. The mathematical unification is a single joint token space $V = V_{text} \cup V_{image} \cup V_{audio} \cup \ldots$ over which a single softmax is computed. The key theoretical question is whether the optimal representations for generation (requiring fine-grained local detail) and understanding (requiring semantic abstraction) can coexist in the same token vocabulary — empirical evidence suggests yes, at sufficient model scale.

Concluding Remarks on Part VI and the Treatise

The eight parts of this treatise have traced a complete arc from the geometric foundations of linear algebra and information geometry, through the mathematical logic of neural network architectures, into the statistical and Bayesian frameworks for learning and uncertainty quantification, the generative paradigm of distribution learning, the sequential decision theory of reinforcement learning and alignment, and finally the advanced mathematical structures — statistical physics, algebraic topology, causal inference, and

scaling theory — that provide the deepest understanding of why deep learning works and where its limits lie.

Three overarching mathematical themes have recurred throughout. First, the geometry of high-dimensional probability distributions: the Fisher metric, optimal transport, the information geometry of exponential families, and the Riemannian structure of the statistical manifold appear in natural gradient descent, WGAN, flow matching, and CLIP alignment alike. Second, the role of symmetry and equivariance: from the Transformer's permutation equivariance to the SE(3)-equivariance of molecular models, symmetry constraints are the primary mechanism by which architectural inductive biases are encoded mathematically. Third, the interplay between overparameterisation and generalisation: double descent, the NTK regime, benign overfitting, and the flat minima hypothesis all address the central mystery of why models with more parameters than data can generalise at all.

These themes do not resolve into a single unified theory — the mathematics of deep learning remains, in 2024, an active frontier with major open questions. But the tools assembled here provide the language in which those questions can be asked precisely, which is the prerequisite for answering them.

PART VII

SCIENTIFIC AI AND GEOMETRIC DEEP LEARNING

Part VII of the Mathematics of Deep Learning

SE(3) Equivariance · AlphaFold 2 · AlphaFold 3 · ESM-3 · AlphaGenome · Scientific Foundation Models

Preface: The Mathematics of Scientific AI

Scientific AI applies deep learning to problems in the natural sciences — protein structure prediction, genomics, drug discovery, quantum chemistry — where the data has rich geometric and physical structure that must be encoded in the architecture. The central mathematical challenge distinguishes this domain from natural language and image processing: biological and chemical objects exist in three-dimensional space and obey physical symmetries that must be respected exactly, not merely approximately. A protein structure prediction model that is not equivariant to rotations and translations in 3D space is not merely sub-optimal — it makes predictions that are physically meaningless.

This part develops the mathematical foundations of scientific AI, beginning with the theory of group equivariance that underlies all geometric deep learning, proceeding through AlphaFold 2 and 3 as the central case study of SE(3)-equivariant architecture design, and closing with ESM-3 and AlphaGenome as examples of biological foundation models that extend these principles to protein language and genomic sequence.

Chapter 1: Groups, Symmetries, and Equivariant Maps

The deep connection between symmetry and the structure of physical laws is expressed mathematically by group theory. Understanding why equivariance is the correct inductive bias for scientific AI — and not merely a useful heuristic — requires the mathematical framework of group representations and equivariant linear layers.

1.1 Symmetry Groups Relevant to Scientific AI

The three primary symmetry groups in scientific AI are:

S_n (Symmetric group): permutation group on n elements. Relevant for any problem where inputs are sets (molecules as sets of atoms, sequences as sets of amino acids or codons). Attention is S_n-equivariant: permuting the input tokens permutes the output tokens accordingly.

SE(3) (Special Euclidean group in 3D): rigid body motions — rotations SO(3) and translations $\mathbb{R}^3$. Relevant for 3D molecular structures: a protein has the same energy, same function, and same fold regardless of how it is oriented or positioned in 3D space. A model learning protein properties must respect this invariance/equivariance.

$SE(3) \times S_n$: for molecules as 3D point clouds — both the assignment of atoms to positions must be permutation-equivariant and the 3D coordinates must be SE(3)-equivariant.

Theorem 1.1 (Schur's Lemma and Equivariant Linear Layers)

Let $\varrho_i: G \to GL(V_i)$ and $\varrho_j: G \to GL(V_j)$ be two irreducible representations (irreps) of a group G. A linear map $L: V_i \to V_j$ is G-equivariant ($L \circ \varrho_i(g) = \varrho_j(g) \circ L$ for all $g \in G$) if and only if:(i) $V_i \cong V_j$ (the representations are isomorphic), in which case $L = \lambda I$ for some scalar λ.(ii) $V_i \not\cong V_j$, in which case $L = 0$.Consequence: The only equivariant linear maps between irreps are scalar multiples of the isomorphism (for equivalent irreps) and zero (for inequivalent irreps). An equivariant linear layer between general representations (direct sums of irreps) is a block-diagonal matrix with each block being a scalar multiple of the identity on matching irreps — far fewer parameters than a general linear layer.

1.2 Irreps of SO(3) and Spherical Harmonics

The irreducible representations of SO(3) are indexed by non-negative integers $\ell \in \mathbb{N}_0$ (the **angular momentum quantum number**) and have dimension $2\ell + 1$. The corresponding basis functions are the **real spherical harmonics** $\{Y_\ell^m\}_{m=-\ell}^{\ell}$: functions on the sphere $S^2 = \{\hat{r} \in \mathbb{R}^3 : \|\hat{r}\| = 1\}$ that transform under rotation $R \in SO(3)$ as:

$$(R \cdot Y_\ell^m)(\hat{r}) = Y_\ell^m(R^{-1}\hat{r}) = \sum_{m'} D^{(\ell)}_{m'm}(R)\, Y_\ell^{m'}(\hat{r})$$

where $D^{(\ell)}(R)$ is the Wigner D-matrix, the $(2\ell+1) \times (2\ell+1)$ matrix representation of R in the ℓ-th irrep. The key properties: $\ell=0$ (scalar, invariant under rotation), $\ell=1$ (vector, transforms as a 3D vector), $\ell=2$ (rank-2 symmetric traceless tensor, 5 dimensions), etc.

1.3 The Clebsch-Gordan Product

Given two equivariant features of angular momenta ℓ_1 and ℓ_2, their **tensor product** decomposes into a direct sum of irreps by the **Clebsch-Gordan decomposition**:

$$V^{\ell_1} \otimes V^{\ell_2} = \bigoplus_{\ell=|\ell_1-\ell_2|}^{\ell_1+\ell_2} V^{\ell}$$

The Clebsch-Gordan coefficients $\langle \ell_1 m_1,\ \ell_2 m_2 \mid \ell m \rangle$ give the explicit decomposition. This is the SO(3)-equivariant analogue of multiplication: the product of two equivariant features is itself equivariant, and its type is determined by the triangle rule $|\ell_1-\ell_2| \le \ell \le \ell_1+\ell_2$. The tensor product is the fundamental operation in SE(3)-equivariant networks (e3nn, SEGNN, NequIP), but is computationally expensive: the cost scales as $O((\ell_{max})^3)$ per interaction. AlphaFold's design choice to avoid spherical harmonics (using frames instead) is motivated by this cost.

Chapter 2: AlphaFold 2 — Architecture and Mathematical Foundations

AlphaFold 2 (Jumper et al., 2021) solved the protein structure prediction problem to near-experimental accuracy, one of the most significant achievements in the history of computational biology. Its architecture is a masterpiece of domain-specific inductive bias encoding: every component reflects a precise mathematical decision about how to represent the evolutionary, chemical, and geometric information relevant to protein structure.

2.1 The Dual Representation: MSA and Pair Features

AlphaFold 2 represents a protein with N residues using two tensors: the **MSA representation** $m \in \mathbb{R}^{S \times N \times d_{msa}}$ (sequences in the multiple sequence alignment, capturing evolutionary information) and the **pair representation** $z \in \mathbb{R}^{N \times N \times d_{pair}}$ (one feature vector per pair of residues, capturing spatial and chemical relationships). The MSA tensor has S rows (homologous sequences) and the pair tensor is a $N \times N$ matrix of vectors.

This dual representation is motivated by a causal model of protein evolution: homologous sequences carry **co-evolutionary signals** (positions that mutate together tend to be in contact in 3D structure), which are most efficiently captured in the MSA. The pair representation directly encodes the output space — 3D distances and orientations between pairs of residues — and is updated to eventually predict the structure.

2.2 The Evoformer: Axial Attention and Triangular Updates

The **Evoformer** block updates the MSA and pair representations through a sequence of operations. The key operations are:

MSA row-wise gated attention with pair bias: each row of the MSA (one sequence) attends to other positions in the same sequence, with attention logits biased by the pair representation:

$$a_{ij} = \mathrm{softmax}((Q_i K_j^T/\sqrt{d}) + b_{ij}) \quad \text{where } b_{ij} \text{ is a linear projection of } z_{ij}$$

MSA column-wise gated attention: each column of the MSA (all sequences at one position) attends to other sequences, capturing co-evolutionary signal.

Outer product mean: updates the pair representation from the MSA by computing the outer product of MSA features at each position pair:

$$z_{ij} \mathrel{+}= \mathrm{mean}_s(m_{is} \otimes m_{js}) \quad \text{(mean over MSA sequences } s\text{)}$$

Theorem 2.1 (Triangular Updates as Geometric Invariants)

The triangular multiplicative update implements a constraint from metric geometry: if A, B, C are three residues, then the distance d(A,C) is approximately determined by d(A,B) and d(B,C) via the triangle inequality. The "outgoing" triangular update: $\Delta z_{ij} = \sigma(g_{ij}) \odot [(a_{ik} \odot b_{kj})]_{\text{agg over k}}$ accumulates evidence about i-j distance from all "bridge" residues k, implementing an

approximate version of the triangle inequality constraint. The two variants (incoming/outgoing) cover both A→B→C and A←B→C patterns. This geometric constraint is the key mechanism by which the pair representation learns a self-consistent 3D distance matrix before the structure module converts it to 3D coordinates.

2.3 Rigid Frames and SE(3)-Equivariance

AlphaFold 2 avoids spherical harmonics by using **rigid frames**: a frame is a rotation matrix $R \in SO(3)$ plus a translation $t \in \mathbb{R}^3$ (i.e., an element of SE(3)) attached to each residue, defined by the backbone atoms N-Cα-C. The N, Cα, and C atoms define three points that uniquely determine a right-handed coordinate frame via Gram-Schmidt orthogonalisation.

An SE(3)-equivariant operation on frames satisfies: $f(g \circ T_1, \ldots, g \circ T_n) = g \circ f(T_1,\ldots,T_n)$ for any $g \in SE(3)$. AlphaFold achieves equivariance via the **Invariant Point Attention (IPA)** mechanism.

2.4 Invariant Point Attention (IPA)

IPA computes attention using both sequence-space features and 3D-space coordinates. For each residue i with frame $T_i = (R_i, t_i)$, the attention logit between i and j is:

$$a_{ij} = (1/\sqrt{(d_h + \gamma_q)}) \, [q_i^T k_j + (\gamma_q/2) \, \Sigma_h \, \| R_i \tilde{q}_{ih} - R_j \tilde{k}_{jh} \|^2]$$

The first term is the standard dot-product attention in feature space. The second term is the sum of squared distances between **query and key points** $\tilde{q}_{ih}$, $\tilde{k}_{jh}$ **expressed in the local frames** R_i, R_j. Because the distance $\| R_i \tilde{q}_{ih} - R_j \tilde{k}_{jh} \|$ is invariant under global SE(3) transformations (rotating the entire protein rigidly changes neither R_i nor R_j relative to each other), IPA is globally SE(3)-invariant while remaining differentiable.

Theorem 2.2 (IPA SE(3) Invariance)

The IPA attention weights a_{ij} are SE(3)-invariant: for any global rotation $R \in SO(3)$ and translation $t \in \mathbb{R}^3$, applying the rigid transformation $g=(R,t)$ to all frames $T_i \to g \circ T_i$ does not change a_{ij}. The IPA output — the attention-weighted sum of

value features — is SE(3)-equivariant with respect to the output frame positions. Proof: the point distances $\|R_i\tilde{q}_{ih} - R_j\tilde{k}_{jh}\|^2$ are invariant under global $g=(R,t)$ since: $\|(RR_i)\tilde{q}_{ih} + t - (RR_j)\tilde{k}_{jh} - t\|^2 = \|R(R_i\tilde{q}_{ih} - R_j\tilde{k}_{jh})\|^2 = \|R_i\tilde{q}_{ih} - R_j\tilde{k}_{jh}\|^2$ (rotation preserves norm).

2.5 The FAPE Loss

The **Frame Aligned Point Error (FAPE)** measures structural accuracy in a way that is independent of the global orientation:

$$\mathscr{L}_FAPE = (1/NT)\,\Sigma_i\,\Sigma_{xp}\,\|\tilde{T}_i^{-1}\,x_p - \hat{T}_i^{-1}\,x_p\|$$

where $\tilde{T}_i$ is the predicted frame for residue i, $\hat{T}_i$ is the true frame, and x_p ranges over all atom positions. The error at each residue is measured in the **local coordinate frame of that residue**: the predicted and true 3D positions of all atoms are first expressed in residue i's predicted and true local frames respectively, then compared. This frame-relative measurement is more informative than global RMSD because it is **residue-local**: errors in early residues do not compound into errors in all subsequent measurements.

Chapter 3: AlphaFold 3 — Diffusion for Biomolecular Structure

AlphaFold 3 (Abramson et al., 2024) extends AlphaFold 2 beyond proteins to all biomolecular systems: DNA, RNA, small molecules, post-translational modifications, and protein-ligand complexes. The key architectural change is replacing the structure module (which predicts torsion angles and assembles the backbone in a deterministic procedure) with a diffusion model that directly operates on 3D atomic coordinates.

3.1 The Pairformer

AlphaFold 3 replaces the Evoformer with the **Pairformer**: a simplified architecture that processes only the pair representation (no MSA, which is replaced by a single-sequence embedding for non-protein inputs). The Pairformer uses: **triangle attention** (same as Evoformer), **triangle multiplicative updates** (same), and **pair transition** (two-layer MLP on each

pair). The single representation s is updated by attention over the pair representation.

3.2 The Diffusion Module

Given the pair and single representations from the Pairformer, the diffusion module learns to denoise 3D atom positions. The forward process adds Gaussian noise to all atom positions: $x_t = x_0 + \sigma_t\ \varepsilon,\ \varepsilon \sim \mathcal{N}(0, I)$. The diffusion network $D_\theta(x_t,\ t,\ s,\ z)$ takes noisy coordinates, the noise level, and the Pairformer embeddings as input and predicts the clean coordinates $\hat{x}_0$

The training objective is:

$$\mathcal{L}_diff = \mathbb{E}_{t,\varepsilon}[\lambda(t) \cdot \|D_\theta(x_0 + \sigma_t\ \varepsilon,\ t,\ s,\ z) - x_0\|^2]$$

The diffusion network must be SE(3)-equivariant: rotating the input coordinates must rotate the output prediction. This is implemented via a specialized **Atom Transformer** that operates directly on 3D coordinates using IPA-style invariant point attention, with the Pairformer embeddings providing the conditioning context.

Theorem 3.1 (Diffusion as Structure Prediction)

The diffusion module learns the conditional distribution $p(x_0 \mid \text{sequence, MSA, templates})$ — the distribution over 3D structures given the biological context. Unlike AlphaFold 2 which predicts a single structure (the mode), AlphaFold 3 can sample multiple structures from this conditional distribution by running multiple reverse diffusion trajectories from different initial noises. This is particularly valuable for intrinsically disordered regions, flexible loops, and multi-conformation proteins, where a single predicted structure is not an accurate representation of the biological reality.

Chapter 4: ESM-3 — A Multimodal Protein Language Model

ESM-3 (Hayes et al., 2024) is a unified foundation model for protein biology that jointly models protein sequence, structure, and function. Unlike AlphaFold (conditioned structure prediction) or protein language models

(sequence modelling only), ESM-3 treats all three modalities as tokenised sequences and trains a single masked language model on their joint distribution.

4.1 Tokenisation of the Three Modalities

Sequence: 20 amino acid tokens + gaps and special tokens. Standard discrete vocabulary.

Structure: 3D coordinates are tokenised via a VQ-VAE trained to quantise structural neighbourhoods. Each residue's local geometric environment (its 3D neighbourhood of ±16 residues) is encoded as a single discrete token from a vocabulary of ~4096 structure tokens. This is the **structure tokeniser**: it converts 3D coordinates into a sequence of discrete symbols that can be modelled autoregressively.

Function: InterPro annotations and Gene Ontology terms are encoded as binary bit vectors (function tokens), one per known functional family.

The three token types are embedded into a shared latent space. A **geometric attention** module processes the structure tokens in a geometry-aware manner, analogous to IPA but adapted for the discrete structure token representation.

4.2 The Unified Masked Language Model

ESM-3 is trained with masked language modelling simultaneously across all three modalities: some sequence positions, some structure tokens, and some function annotations are masked, and the model predicts the masked values from the unmasked context. The key property is **cross-modal conditioning**: when structure is provided (unmasked) but sequence is masked, the model performs **inverse folding** (predicting sequence from structure); when sequence is provided but structure is masked, the model performs **structure prediction** (similar to AlphaFold but generative).

Theorem 4.1 (Generative Capacity of Multimodal MLM)

Let S, T, F denote sequence, structure, and function tokens respectively. ESM-3 trained as a joint masked LM learns the joint distribution $p(S, T, F)$ and all conditionals $p(S|T,F)$, $p(T|S,F)$, $p(F|S,T)$, $p(S|T)$, etc. simultaneously. Any conditional can be evaluated by setting the appropriate modalities as context

(unmasked) and computing the masked token probabilities. This enables:— Structure prediction: sample p(T|S)— Inverse folding: sample p(S|T)— Function prediction: sample p(F|S)— Unconditional generation: sample p(S,T,F)all from a single trained model, without task-specific fine-tuning.

Chapter 5: AlphaGenome — Deep Learning for the Genome

AlphaGenome (Benegas et al., 2024) applies the lessons of AlphaFold to genomics: predicting multiple regulatory phenotypes — gene expression, chromatin accessibility, transcription factor binding, chromosome topology — from raw DNA sequence. The key mathematical challenges are the extreme length of genomic sequences (up to 1 Mb), the multi-scale structure of genomic regulation (from single nucleotides to chromosome-scale domains), and the Watson-Crick complementarity symmetry of double-stranded DNA.

5.1 Multi-Scale Genomic Tokenisation

Raw DNA sequences of length $L = 10^5–10^6$ nucleotides are processed through a **multi-scale convolution stem** that progressively downsamples the sequence while extracting features at multiple resolutions:

$$h^{(1)} = Conv_{k=15}(x_{nuc}),\ \ dim:\ L \to L\ (nucleotide\ resolution)$$

$$h^{(2)} = Pool + Conv(h^{(1)}),\ \ dim:\ L \to L/4$$

$$h^{(k)} = Pool + Conv(h^{(k-1)}),\ \ dim:\ L/4^{k-1} \to L/4^k$$

The resulting multi-scale feature maps are processed by **cross-scale attention** that allows each scale to attend to information from other scales, enabling long-range genomic interactions (e.g., enhancer-promoter contacts at distances of 100kb) to influence local regulatory predictions.

5.2 Watson-Crick Complementarity as a Symmetry

Double-stranded DNA has a fundamental symmetry: the complement of the forward strand is the reverse complement of the reverse strand. For any sequence $x = x_1x_2...x_l$, its reverse complement is $RC(x) = c(x_l)c(x_{\ell-1})...c(x_1)$ where

$c(A)=T$, $c(T)=A$, $c(C)=G$, $c(G)=C$. Regulatory function is symmetric under this operation: the set of transcription factors binding a sequence is the same as those binding its reverse complement.

AlphaGenome enforces this symmetry via **reverse complement equivariance**: the model is trained on both the forward and reverse complement sequences, and the outputs are averaged. This is equivalent to constructing a model where $f(RC(x)) = RC(f(x))$ by architecture design. The weight sharing induced by this symmetry reduces the effective parameter count by roughly 2× and improves sample efficiency.

5.3 Multi-Task Prediction and Gradient Conflict

AlphaGenome simultaneously predicts k regulatory phenotypes from shared representations: RNA-seq (gene expression), ATAC-seq (chromatin accessibility), ChIP-seq (TF binding at n_{TF} TF sites), Hi-C (3D chromosome contacts), and splicing ratios. The multi-task loss is:

$$\mathcal{L} = \Sigma_t \lambda_t \mathcal{L}_t, \quad \lambda_t \propto 1/\| \nabla_\theta \mathcal{L}_t \|$$

where task weights are set to equalise gradient magnitudes (approximate GradNorm). The fundamental challenge is **gradient conflict**: tasks whose optima lie in conflicting directions of parameter space. PCGrad (Yu et al., 2020) projects conflicting gradients:

$$\hat{g}_t = g_t - (g_t \cdot g_t'/\|g_t'\|^2) g_t' \quad \text{if } g_t \cdot g_t' < 0$$

eliminating the component of g_t that conflicts with g_t', reducing negative interference between tasks with misaligned gradients.

Chapter 6: Equivariant GNNs for Molecular Property Prediction

Beyond protein structure prediction, SE(3)-equivariant graph neural networks are the state of the art for predicting molecular properties from 3D structure: drug-target binding affinity, reaction rates, material properties. This chapter develops the mathematical framework that connects the group-theoretic foundations of Chapter 1 to practical architectures.

6.1 SchNet, DimeNet, and the Progressive Addition of Geometric Information

The progression from SchNet to DimeNet to PaiNN illustrates how successive incorporation of geometric information improves expressivity:

SchNet (Schütt et al., 2017): uses only pairwise distances $d_{ij} = \|r_i - r_j\|$ (invariant scalars). Message: $m_{ij} = \varphi(h_i, h_j, d_{ij})$. Cannot distinguish mirror images (chirality).

DimeNet (Klicpera et al., 2020): uses distances and angles θ_{jik} (invariant under SE(3)). Message incorporates the triplet $(d_{ij}, d_{jk}, \theta_{jik})$. Can distinguish some non-isomorphic graphs that SchNet cannot.

PaiNN (Schütt et al., 2021): uses equivariant vector features alongside scalar features. Each atom has both a scalar state $s_i \in \mathbb{R}^d$ and a vector state $v_i \in \mathbb{R}^{3\times d}$ that transforms as a 3D vector. Equivariant messages update vector features:

$$\Delta v_i = \Sigma_j \varphi_v(s_i, s_j, d_{ij}) \cdot (r_j - r_i)/d_{ij}$$

Theorem 6.1 (Expressivity Hierarchy of Geometric GNNs)

For molecular graphs with 3D coordinates, the following strict expressivity hierarchy holds: Distance-only (SchNet) $\subsetneq$ Angle-augmented (DimeNet) $\subsetneq$ Vector-equivariant (PaiNN) $\subsetneq$ Tensor-equivariant (NequIP, MACE)Specifically: (i) there exist pairs of molecules indistinguishable by SchNet but distinguishable by DimeNet; (ii) there exist chiral pairs indistinguishable by DimeNet but distinguishable by PaiNN; (iii) there exist molecules where accurate force prediction (directional, equivariant) requires tensor-equivariant architectures. The cost of higher expressivity is $O((\ell_{max})^3)$ per tensor product operation, motivating efficient approximations like MACE's body-order expansion.

Concluding Remarks on Part VII

Scientific AI represents the most mathematically demanding application of deep learning: the architectures must respect the physical laws that govern biological and chemical systems, and these laws are expressed in the language of group theory, differential geometry, and statistical mechanics. The

progression from the general theory of equivariant maps (Chapter 1) through AlphaFold's engineering of SE(3) equivariance via rigid frames and IPA (Chapters 2-3) to the multimodal foundation models of ESM-3 and AlphaGenome (Chapters 4-5) illustrates a general principle: the most powerful scientific AI systems are those where the inductive bias of the architecture precisely encodes the symmetry structure of the domain.

The connection to other parts of this treatise is deep. The information-geometric framework of Part I appears in the structure of AlphaFold's confidence metrics (pLDDT and pTM); the diffusion model framework of Part IV reappears in AlphaFold 3's structure prediction module; the multi-task learning theory connects to the gradient conflict analysis of Part III; and the scaling laws of Part VI predict that further increases in training data and model size will continue to improve scientific AI systems substantially beyond current performance.

PART VIII

AGENTIC AI, META-LEARNING, AND CONTINUAL LEARNING

Part VIII of the Mathematics of Deep Learning

Agentic LLMs · MARL · Inference-Time Compute · MAML · Video Generation · Continual Learning · Topological Analysis

Preface: Beyond Single-Step Prediction

The preceding parts of this treatise have addressed systems that produce a single output given a single input — a classification, a generated image, a predicted structure. The frontier of deep learning increasingly concerns systems that must act over extended time horizons, adapt to new tasks from few examples, maintain competence across a stream of changing

environments, and coordinate with other agents. These challenges require mathematical frameworks beyond standard supervised and generative learning: the theory of partially observable decision processes, the game-theoretic analysis of multi-agent systems, the meta-learning formulation of task distributions, and the continual learning framework for non-stationary distributions.

This part assembles the mathematics of these frontiers. Chapter 1 develops the POMDP framework for agentic LLMs and the game theory of multi-agent systems. Chapter 2 addresses inference-time compute scaling: MCTS, process reward models, and the GRPO algorithm. Chapter 3 extends diffusion and flow matching to video. Chapter 4 develops the theory of meta-learning with MAML as the central case. Chapter 5 addresses catastrophic forgetting and continual learning. Chapter 6 closes with the topological analysis of learning dynamics.

Chapter 1: Agentic LLMs and Multi-Agent Reinforcement Learning

Large language models deployed as autonomous agents — systems that perceive observations, execute tools, reason over multi-step plans, and interact with other agents — require a mathematical framework that extends beyond standard language modelling. The POMDP formalism provides this framework, and multi-agent extensions introduce game-theoretic considerations absent in single-agent settings.

1.1 The ReAct Framework as a POMDP

A **Partially Observable Markov Decision Process (POMDP)** is a tuple $(S, A, O, P, R, \Omega, \gamma)$ where S is the (hidden) state space, A the action space, O the observation space, $P: S\times A\to\Delta(S)$ the transition kernel, $R: S\times A\to\mathbb{R}$ the reward function, $\Omega: S\times A\to\Delta(O)$ the observation model, and γ the discount factor. The agent does not observe $s\in S$ directly but receives an observation $o\in O$ and must act based on its **belief state** — the posterior $b(s) = P(S=s \mid o_1,a_1,\ldots,o_t)$.

The ReAct framework (Yao et al., 2022) instantiates LLM agents as POMDPs where: the hidden state S is the world state (e.g., contents of a web page, database); observations O are textual tool outputs (search results, code

execution outputs); actions A are generated text strings (tool calls, reasoning steps, final answers); and the LLM's context window serves as an explicit **belief representation** — the concatenation of observations and actions up to time t constitutes the agent's belief state.

Theorem 1.1 (Context Window as Belief State)

For an LLM agent with context window $C = (o_1, a_1, \ldots, o_t, a_t)$, the conditional distribution $\pi_LLM(a_{t+1} \mid C)$ is a valid POMDP policy if and only if the context window is a sufficient statistic of the belief state — i.e., $P(s_t \mid C) = P(s_t \mid b(C))$. This holds under the Markov assumption on the world state but fails when (i) observations are informative about states outside the context window (long-horizon dependencies) or (ii) the context is truncated (finite context length). The failure mode is precisely long-horizon reasoning failure: the agent's policy cannot condition on relevant past observations that have been removed from the context, making it a suboptimal POMDP policy.

1.2 Tool Use and the Grounding Problem

LLM agent actions include **tool calls**: structured outputs that invoke external functions (web search, code execution, API calls) and receive structured responses. The **grounding problem** asks how to ensure that the LLM's internal representations of world state accurately reflect the actual state after tool execution.

Formally, let $T: A_tools \times S \to O \times S$ be the tool execution function. The agent policy must reason about $P(s_{t+1} \mid o_{t+1}, T(a_t, s_t))$ — the updated belief after observing the tool output. Grounding fails when the LLM hallucinates tool outputs inconsistent with T or when it ignores tool outputs in favour of its prior. The mathematical solution is **observation grounding**: explicitly inserting tool outputs into the context at the correct position and training the model to assign high probability to the correct next token given these observations, which requires specialised training on tool-augmented trajectories.

1.3 Multi-Agent Reinforcement Learning

In a multi-agent system, N agents with policies $\pi = (\pi_1,\ldots,\pi_n)$ interact in a shared environment. The **joint policy** π induces a distribution over trajectories and each agent i has its own return $J_i(\pi)$. The three canonical cooperative/competitive settings are:

Cooperative: all agents share a reward, $R_1 = \ldots = R_n = R$. The team-optimal joint policy maximises $J(\pi) = \mathbb{E}[\Sigma_t\, \gamma^t\, R(s_t,a_t)]$.

Zero-sum: $\Sigma_i\, R_i = 0$ at every step. The solution concept is the Nash equilibrium.

General-sum: arbitrary reward functions. Both Nash equilibria and correlated equilibria are solution concepts.

Theorem 1.2 (Nash Equilibrium Existence)

In any finite N-player normal form game (finite action spaces, general-sum rewards), at least one Nash equilibrium in mixed strategies exists. Proof: the best-response correspondence BR_i(π_{-i}) = argmax_{πᵢ} Jᵢ(πᵢ, π_{-i}) is upper-hemicontinuous and convex-valued on the compact mixed strategy simplex. By Kakutani's fixed-point theorem, the joint best-response correspondence BR(π) = ×ᵢ BRᵢ(π_{-i}) has a fixed point π* = BR(π*) — which is precisely a Nash equilibrium. Note: the existence theorem does not imply uniqueness, computability in polynomial time, or convergence of RL algorithms to Nash equilibria.

1.4 QMIX and MAPPO: Cooperative MARL Algorithms

Independent Q-Learning (IQL) trains each agent's Q-function independently, ignoring other agents. This fails because from agent i's perspective, other agents are non-stationary — their policies change during training, violating the stationarity assumption required for Q-learning convergence.

QMIX (Rashid et al., 2018) addresses this with centralised training and decentralised execution: a **mixing network** $Q_{tot}(\tau,\ a) = f_{mix}(Q_1(\tau_1,a_1),\ldots,Q_n(\tau_n,a_n))$ aggregates individual Q-values into a global value. The mixing network satisfies the **Individual-Global-Max (IGM) condition**:

$$\operatorname{argmax}_a\, Q_{tot}(\tau,a) = \left(\operatorname{argmax}_{a_1} Q_1(\tau_1,a_1),\ \ldots,\ \operatorname{argmax}_{a_n} Q_n(\tau_n,a_n)\right)$$

This requires $\partial Q_{tot}/\partial Q_i \geq 0$ (monotonicity), enforced by making all mixing network weights non-negative. Under IGM, each agent can act greedily with

respect to its own Q_i without coordination, yet the joint action is globally optimal for Q_{tot}.

MAPPO applies PPO to cooperative MARL with centralised value functions: the critic $V_\varphi(s)$ has access to the global state s (available during training) while each actor $\pi_i(a_i \mid o_i)$ uses only local observations. The centralised critic enables variance reduction via accurate baselines without violating the decentralised execution constraint.

1.5 Mean-Field Games and the Many-Agent Limit

For systems with $N \to \infty$ homogeneous agents, Mean-Field Game (MFG) theory (Lasry and Lions, 2007) provides a tractable approximation. In the mean-field limit, each agent interacts not with specific other agents but with the **population distribution** (mean field) $\mu_t \in \Delta(S)$ of all agents. The MFG equilibrium is a pair (π^*, μ^*) satisfying two coupled equations:

Forward equation: given policy π^*, the population evolves as $\mu_{t+1} = T(\mu_t, \pi^*)$

Backward equation: given distribution μ^*, the optimal policy is $\pi^* = \text{argmax}_\pi J_i(\pi, \mu^*)$

The fixed point (π^*, μ^*) exists under mild conditions (Lipschitz rewards, compact state/action spaces) and can be computed by iterating the forward-backward system. MFGs are tractable because the N^2-dimensional interaction space is compressed to the finite-dimensional space of distributions $\Delta(S)$. Applications include traffic routing, financial market modelling, and swarm behaviour.

Chapter 2: Inference-Time Compute Scaling

The Kaplan and Chinchilla scaling laws describe how model quality improves with training compute. A complementary scaling dimension — test-time compute — has emerged as equally important: allocating more compute at inference time (through tree search, iterative refinement, or sampling with verification) can recover the quality gap between smaller and larger models, with implications for the design of AI systems.

2.1 Monte Carlo Tree Search for LLMs

MCTS (Coulom, 2006) is a best-first tree search algorithm that balances exploration and exploitation via the UCB1 selection rule. Applied to LLM decoding, each tree node is a partial generation (token sequence), and each edge is a token extension. The UCB1 selection rule at node v selects child $c^*(v)$:

$$c^*(v) = \text{argmax}_c \left[Q(c)/N(c) + C\sqrt{(\log N(v)/N(c))} \right]$$

where $Q(c)$ is the total reward accumulated through node c, $N(c)$ its visit count, and C the exploration constant. The MCTS loop: **selection** (follow UCB1 to a leaf), **expansion** (generate candidate children by sampling from the LLM), **simulation** (roll out to completion with a rollout policy), **backpropagation** (update Q and N along the selected path).

Theorem 2.1 (UCB1 Regret Bound)

For a bandit problem with K arms and T rounds, UCB1 achieves expected cumulative regret: $\mathbb{E}[R_t] \leq \Sigma_i: \Delta_i>0 \left[8 \log(T)/\Delta_i + (1 + \pi^2/3)\Delta_i \right]$where $\Delta_i = \mu^* - \mu_i$ is the gap between the optimal arm mean μ^* and arm i. This is $O(\sqrt{(KT \log T)})$ total regret, matching the information-theoretic lower bound up to logarithmic factors. In the MCTS context, this means UCB1 exploration guarantees that the tree search converges to the optimal action at a rate governed by the reward gap Δ_i — actions close to optimal are explored often, clearly suboptimal actions are quickly abandoned.

2.2 Process Reward Models

Outcome Reward Models (ORMs) score complete solutions. **Process Reward Models (PRMs)** score each step of a reasoning chain, providing denser signal. A PRM assigns a score $r(x, s_1,\ldots,s_k)$ to the partial solution $(s_1,\ldots,s_k)$ given problem x, trained on human annotations of step correctness.

The PRM can be interpreted as estimating the **value function** of the reasoning process: $V_PRM(x, s_1,\ldots,s_k) \approx P(\text{correct final answer} \mid x, s_1,\ldots,s_k)$ — the probability that the current partial solution will lead to a correct answer. This connection to RL value functions is not merely analogical: the PRM satisfies a Bellman-like consistency condition:

$$V_PRM(x, s_1,\ldots,s_k) = \Sigma_\{s_{k+1}\} \, \pi(s_{k+1} \mid x, s_1,\ldots,s_k) \cdot V_PRM(x, s_1,\ldots,s_{k+1})$$

A perfect PRM would enable optimal MCTS: at each expansion step, rather than rolling out to completion, the PRM provides an immediate value estimate for each candidate extension, making MCTS computationally feasible for generation tasks where full rollouts are expensive.

2.3 Best-of-N with Verifier and Scaling Laws

Best-of-N sampling generates N candidate solutions, scores them with a verifier (ORM or PRM), and returns the highest-scoring solution. The accuracy of Best-of-N scales predictably with N:

$$P(\text{best-of-}N \text{ correct}) = 1 - (1-p)^N$$

where p is the per-sample accuracy. For $p = 0.1$ and $N = 100$, this gives $1 - 0.9^{100} \approx 0.99997$ — near-perfect accuracy from a model with 10% per-sample accuracy. The compute cost is $O(N)$ LLM evaluations plus $O(N)$ verifier evaluations. The scaling law $P \propto 1-(1-p)^N$ establishes that test-time compute scales **exponentially efficiently** for problems with independent errors: doubling N raises the error probability from $(1-p)^N$ to $(1-p)^{2N} = ((1-p)^N)^2$, squaring the remaining failure probability.

2.4 GRPO: Group Relative Policy Optimisation

GRPO (Shao et al., 2024) is the RL algorithm underlying DeepSeek-R1 and similar reasoning models. It replaces the PPO critic with a **group-relative baseline**: for each problem x, sample G responses $\{y_1,\ldots,y_G\}$ from the current policy and compute the normalised advantage:

$$\hat{A}_i = \frac{r(x_i,y_i) - \text{mean}(\{r(x_i,y_j)\}_j)}{\text{std}(\{r(x_i,y_j)\}_j)}$$

The GRPO objective is the PPO-clipped objective with this group-relative advantage:

$$\mathcal{L}_{GRPO}(\theta) = \mathbb{E}_{x,\{y_i\}\sim\pi_{\theta old}}\Big[\sum_i \min\big(r_i(\theta)\hat{A}_i,\ \text{clip}(r_i(\theta),1-\varepsilon,1+\varepsilon)\hat{A}_i\big) - \beta D_{KL}(\pi_\theta \| \pi_{ref})\Big]$$

where $r_i(\theta) = \pi_\theta(y_i|x)/\pi_{\theta old}(y_i|x)$. GRPO eliminates the value network entirely — saving ~50% of training memory and compute — while maintaining the variance reduction of advantage-based gradients via the within-group normalisation. The group-relative baseline is an unbiased estimator of the advantage under the policy: $\mathbb{E}_{y\sim\pi}[\hat{A}] = 0$ by construction.

Chapter 3: Video Generation and Spatiotemporal Models

Video generation extends image diffusion models to the temporal dimension, introducing new challenges: the 4D nature of video data (spatial × temporal), the long-range temporal dependencies between frames, the computational cost of attention over spatiotemporal sequences, and the physical consistency constraints that videos must satisfy.

3.1 Video as a 4D Spatiotemporal Signal

A video clip $v \in \mathbb{R}^{T\times H\times W\times C}$ (T frames, H×W pixels, C channels) lives in a much higher-dimensional space than a single image. The key challenges are:

Temporal consistency: adjacent frames must be coherent — objects must move smoothly, lighting must change gradually, physical laws must be respected.

Computational cost: full 3D self-attention has complexity $O((THW)^2)$. For $T=16$, $H=W=64$: sequence length 65,536 and attention cost 4.3 billion operations — prohibitive.

Temporal entanglement: models must learn both spatial appearance (what things look like) and temporal dynamics (how they move), two distinct aspects that ideally should be factored.

3.2 Factored Spatiotemporal Attention

Video Diffusion Models (Ho et al., 2022) address the computational challenge via **factored attention**: separate spatial attention (each frame attending within itself) and temporal attention (each spatial position attending across frames):

$$\text{Spatial: } a^{\text{spatial}}_{(t,i),(t,j)} = \text{softmax}(Q_{(t,i)} K_{(t,j)}^T/\sqrt{d}) \quad \text{[same frame } t\text{]}$$

$$\text{Temporal: } a^{\text{temporal}}_{(t,i),(t',i)} = \text{softmax}(Q_{(t,i)} K_{(t',i)}^T/\sqrt{d}) \quad \text{[same position } i\text{, different frames]}$$

Factored attention reduces complexity from $O((THW)^2)$ to $O(T(HW)^2 + (HW)T^2)$. For $T=16$, $H=W=64$: 2.7×10^8 — a 15× reduction. The factored approach loses some cross-frame-cross-spatial interactions, but empirically the independence

assumption between spatial and temporal processing is a good approximation for natural videos.

Theorem 3.1 (Expressive Power of Factored vs Full Attention)

Full spatiotemporal attention can represent any function of the video tensor v ∈ ℝ^{T×H×W×d}. Factored attention (spatial followed by temporal) can represent any function that factors as f(v) = g(h(v_{spatial}), v_{temporal}) where h aggregates spatial information and g combines spatial aggregates across time. The set of functions representable by factored attention is a strict subset of those representable by full attention: there exist video processing tasks (e.g., detecting diagonal spatiotemporal patterns) that require full attention. Empirically, natural video generation does not appear to require these extra-capacity functions, explaining why factored models achieve competitive quality.

3.3 3D RoPE and the DiT Architecture for Video

Sora (Brooks et al., 2024) is based on the **Diffusion Transformer (DiT)** architecture with a spatiotemporal extension. Videos are tokenised as $p_t \times p_H \times p_W$ spatiotemporal patches (e.g., 1×8×8 pixels per patch). Position encodings use **3D RoPE** — extending the 1D rotary position embedding of transformers to three dimensions by assigning separate frequency components to temporal, height, and width axes:

$$RoPE_3D(q_i, (t,h,w)) = R_t(t)\, R_h(h)\, R_w(w)\, q_i$$

where R_t, R_h, R_w are 2D rotation matrices in different subspaces of the embedding. 3D RoPE preserves the relative position property — the dot product $\langle R(t,h,w)q, R(t',h',w')k \rangle$ depends only on the relative offsets $(t-t', h-h', w-w')$ — crucial for generalisation to videos of different lengths and resolutions.

3.4 Flow Matching for Video Generation

Flow matching (Chapter 7 of Part IV) extends naturally to video: the vector field $v_\theta(x_t, t): \mathbb{R}^{T\times H\times W\times C} \times [0,1] \to \mathbb{R}^{T\times H\times W\times C}$ generates a flow from Gaussian noise to the video distribution. The conditional flow matching loss with optimal transport couplings:

$$\mathcal{L}_{CFM}(\theta) = \mathbb{E}_{t,x_0,x_1}[\| v_\theta(\varphi_t(x_0,x_1), t) - (x_1 - x_0) \|^2_F]$$

where $\|\cdot\|_F$ is the Frobenius norm over all spatiotemporal dimensions. The advantage over diffusion for video is that flow matching trajectories are straighter (by the OT coupling), requiring fewer sampling steps — critical for video where the high-dimensional output makes sampling expensive. CogVideoX and similar models use flow matching with 3D full attention, demonstrating high-quality text-to-video synthesis in 50 sampling steps.

Chapter 4: Meta-Learning — Learning to Learn

Meta-learning addresses the few-shot learning problem: given a task distribution, learn a prior that enables rapid adaptation to new tasks from very few examples. The mathematical framework distinguishes between the meta-learning phase (across many tasks) and the adaptation phase (within a single new task), and different algorithms make different assumptions about the structure of the adaptation.

4.1 The Meta-Learning Formulation

Let $p(\mathcal{T})$ be a distribution over tasks, each task $\mathcal{T}_i = (\mathcal{D}_i^{train}, \mathcal{D}_i^{test}, \mathcal{L}_i)$ consisting of a support set, query set, and loss function. The meta-learning objective is:

$$\min_\theta \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})}[\mathcal{L}_{\mathcal{T}}(f_{Adapt(\theta, \mathcal{D}^{train}_{\mathcal{T}})}; \mathcal{D}^{test}_{\mathcal{T}})]$$

The **Adapt** function maps the meta-parameters θ and task support data to task-specific parameters. Different meta-learning algorithms correspond to different parametric forms of Adapt:

MAML: gradient steps from θ on support data

ProtoNets: compute class prototypes from support embeddings

Meta-SGD: learn both initial parameters and per-parameter learning rates

4.2 MAML: Model-Agnostic Meta-Learning

MAML (Finn et al., 2017) sets $Adapt(\theta, \mathcal{D}^{train}) = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{D}^{train}}(f_\theta)$ — one (or K) gradient steps on the support set. The meta-objective becomes:

$$\min_\theta \Sigma_i \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha\nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{train}(f_\theta)}) \quad \text{(meta-loss on query sets)}$$

The meta-gradient requires differentiating through the adaptation gradient step — computing the Hessian of the task loss with respect to θ:

$$\nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{test}(\theta_i') = (I - \alpha \nabla^2_\theta \mathcal{L}_{\mathcal{T}_i}^{train}(\theta)) \nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}^{test}(\theta_i')$$

Theorem 4.1 (MAML as Implicit Gradient Learning)

MAML with K gradient steps finds parameters θ such that, after K adaptation steps on any task $\mathcal{T}$ from $p(\mathcal{T})$, the adapted parameters $\theta_{\mathcal{T}} = \theta - \alpha \Sigma_k \nabla \mathcal{L}$ are near the task optimum. The meta-gradient implicitly computes second-order information: it evaluates how a perturbation of θ propagates through K gradient steps to affect test performance. The MAML optimum θ^* is characterised by: $\nabla_\theta \mathbb{E}_{\mathcal{T}}[\mathcal{L}_{\mathcal{T}}^{test}(\theta^*_{\mathcal{T}})] = 0$, where $\theta^*_{\mathcal{T}}$ is the result of K adaptation steps from θ^*. This is a fixed-point condition: the meta-parameters are optimal if every task-adapted version is at or near its task optimum.

4.3 First-Order Approximations: Reptile and FOMAML

Computing the full MAML meta-gradient requires the Hessian of the task loss, which costs $O(P^2)$ memory. Two first-order approximations avoid this:

FOMAML ignores the second-order terms in the meta-gradient, using only $\nabla_{\theta_i'} \mathcal{L}_{\mathcal{T}_i}^{test}(\theta_i')$ as the meta-gradient. Empirically, FOMAML achieves 95%+ of full MAML performance at a fraction of the cost.

Reptile (Nichol et al., 2018) updates $\theta \leftarrow \theta + \varepsilon(\theta_i' - \theta)$ — moving θ toward the task-adapted parameters. This avoids any second-order computation. Reptile can be shown to approximate MAML in the first-order limit: it implicitly maximises within-task generalisation (the correlation between gradients on different minibatches of the same task), which is equivalent to MAML when the task distribution has a particular structure.

4.4 In-Context Learning vs MAML: Two Meta-Learning Paradigms

Large language models perform meta-learning via in-context learning (ICL): few-shot examples in the context serve as the support set, and the model's next-token prediction is the adapted prediction. The connection to MAML is:

ICL performs meta-learning through the forward pass (attention over context); MAML performs it through the backward pass (gradient descent on support data).

Akyürek et al. (2023) formalise this: a transformer trained on *(x,y)* pairs implements one step of gradient descent on the in-context examples via its attention mechanism, with *W_K^T W_Q* playing the role of the step size times the feature kernel. This makes transformer ICL a special case of kernel regression adaptation, with the context length playing the role of the support set size and the attention pattern implementing the kernel smoother.

Chapter 5: Continual and Lifelong Learning

Continual learning addresses the challenge of learning a sequence of tasks without forgetting previous ones — a fundamental limitation of standard neural network training, which exhibits catastrophic forgetting: gradient steps on new tasks overwrite the parameters relevant to old tasks. The mathematical framework connects Bayesian inference, the geometry of the loss landscape, and the information theory of parameter compression.

5.1 Catastrophic Forgetting: Geometric Analysis

Consider training on task $\mathcal{T}_1$ to convergence at θ_1^*, then training on task $\mathcal{T}_2$. The gradient update $\Delta\theta = -\alpha\nabla_\theta \mathcal{L}_{\mathcal{T}_2}(\theta)$ moves θ in a direction that minimises $\mathcal{L}_{\mathcal{T}_2}$ but may increase $\mathcal{L}_{\mathcal{T}_1}$. A second-order approximation around θ_1^* gives the change in task-1 loss:

$$\Delta L_{\mathcal{T}_1} \approx \Delta\theta^T \nabla^2_\theta \mathcal{L}_{\mathcal{T}_1}(\theta_1^*)\,\Delta\theta = \Delta\theta^T H_1 \Delta\theta$$

where $H_1 = \nabla^2 \mathcal{L}_{\mathcal{T}_1}$ is the Hessian of task 1's loss at θ_1^*. Forgetting is large when $\Delta\theta$ has large components along the high-curvature eigenvectors of H_1 — i.e., when the new task's gradient steps move θ in directions that matter for task 1.

5.2 Elastic Weight Consolidation (EWC)

EWC (Kirkpatrick et al., 2017) addresses forgetting by adding a regularisation term that penalises movement along dimensions important for previous tasks. The key insight is that the Fisher Information Matrix $F_1 = \mathbb{E}_{p_{data,1}}[\nabla_\theta \log p_\theta(y|x)\, \nabla_\theta \log p_\theta(y|x)^T]$ approximates the Hessian H_1 at θ_1^* (the equality $F = -\mathbb{E}[\nabla^2_\theta \log p_\theta]$ holds exactly; the approximation is $F \approx H$ at a maximum likelihood point).

The EWC objective for learning task $\mathscr{T}_2$ while retaining task $\mathscr{T}_1$ is:

$$\mathscr{L}_{EWC}(\theta) = \mathscr{L}_{\mathscr{T}_2}(\theta) + (\lambda/2)\,\Sigma_i\, F_{ii}(\theta_i - \theta_1^{*}{}_i)^2$$

where F_{ii} is the i-th diagonal element of the Fisher matrix. The quadratic penalty $F_{ii}(\theta_i - \theta_1^{*}{}_i)^2$ elastically anchors each parameter θ_i to its task-1 optimum with stiffness λF_{ii} — proportional to how important that parameter is for task 1 (high Fisher diagonal = high importance).

Theorem 5.1 (EWC as Laplace Approximation)

EWC minimises the negative log-posterior $p(\theta \mid \mathscr{D}_1, \mathscr{D}_2)$ where the posterior is approximated as: $\log p(\theta \mid \mathscr{D}_1, \mathscr{D}_2) \approx \log p(\theta \mid \mathscr{D}_2) + \log p(\theta \mid \mathscr{D}_1)$, and $\log p(\theta \mid \mathscr{D}_1)$ is approximated by the Laplace approximation around θ_1^*: $\log p(\theta \mid \mathscr{D}_1) \approx -\tfrac{1}{2}(\theta-\theta_1^*)^T\, F_1\, (\theta-\theta_1^*)$. The EWC objective is the MAP estimator under this approximation. The approximation is exact when p_θ is an exponential family (Fisher = Hessian exactly) and approximate otherwise (curvature of the loss ≈ Fisher information at a maximum). The diagonal approximation of F_1 introduces further approximation error but reduces storage from $O(P^2)$ to $O(P)$.

5.3 Gradient Episodic Memory (GEM) and A-GEM

EWC approximates the constraint "do not increase task-1 loss" with a quadratic penalty. **GEM** (Lopez-Paz and Ranzato, 2017) enforces this constraint exactly. GEM stores an episodic memory $\mathscr{M}_t = \{(x_i, y_i)\}_i$ of examples from each past task and requires that gradient updates do not increase the loss on any stored example:

$$\min_{\Delta\theta} \|\Delta\theta - g\|^2 \quad \text{s.t.} \quad \langle \Delta\theta, g_{t'}, i\rangle \geq 0 \;\; \forall (t', i): t' < t$$

where g is the current task's gradient and $g_{t',i}$ is the gradient on memory example (t', i). The constraint $\langle \Delta\theta, g_{t'} \rangle \geq 0$ ensures that the update does not increase the loss in the direction of past task gradients. The constrained QP is solved by projecting the current gradient g onto the constraint set — feasible if and only if the current and past gradients are not too conflicting. **A-GEM** (Chaudhry et al., 2019) approximates GEM by averaging memory gradients ($g_{ref} = \text{mean}_{t',i}[g_{t',i}]$) and projecting onto a single averaged constraint, reducing per-update cost from $O(|\mathscr{M}|P)$ to $O(P)$

5.4 Progressive Neural Networks and Synaptic Intelligence

Progressive Neural Networks (PNN, Rusu et al., 2016) allocate a new column of parameters for each new task and connect it to all previous columns via lateral connections. The previous columns are frozen, guaranteeing zero forgetting by construction. The lateral connections $h_{l i} = f_l(W_{l i} h_{l-1 i} + \Sigma_{j<i} U_{l j i}\, h_{l-1 j})$ allow the new task to leverage features learned by previous tasks (forward transfer) without modifying them (no forgetting). The cost is linear parameter growth: n tasks require n columns, limiting scalability.

Synaptic Intelligence (SI, Zenke et al., 2017) is an online EWC variant that accumulates importance weights ω_i during training (rather than computing the Fisher at convergence):

$$\omega_i = \Sigma_t\ (-\partial \mathscr{L} / \partial \theta_i \cdot \Delta\theta_i) \,/\, (\Delta\theta_i^2 + \xi)$$

The numerator measures the contribution of parameter i to the loss decrease during training; the denominator normalises by the total parameter change. SI provides online importance estimation, enabling continual learning in streaming settings where tasks do not have clear boundaries.

Chapter 6: Topological Analysis of Learning Dynamics

Topological data analysis provides tools to characterise the geometry of neural network representations and loss landscapes that complement spectral and statistical methods. This chapter applies persistent homology and related tools to understand learning dynamics, grokking, and the topology of feature spaces.

6.1 Simplicial Complexes and the Rips Complex

Given a point cloud $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, the **Vietoris-Rips complex** at scale $\varepsilon > 0$ is the simplicial complex $VR_\varepsilon(X)$ containing all subsets $\sigma \subseteq X$ with diameter $\max_{x,y \in \sigma} d(x,y) \leq \varepsilon$. The Rips complex is nested: $VR_\varepsilon \subseteq VR_{\varepsilon'}$ for $\varepsilon \leq \varepsilon'$, forming a filtration. The homology groups $H_k(VR_\varepsilon(X))$ capture: H_0 = connected components, H_1 = loops, H_2 = enclosed voids.

6.2 Persistence Diagrams as Learning Metrics

The **persistence diagram** D_k of a filtration is a multiset of birth-death pairs $\{(b_i, d_i)\}$ encoding the topological features at each scale. The **bottleneck distance** between two persistence diagrams:

$$d_B(D, D') = \inf_{\gamma: D \to D'} \sup_{p \in D} \|p - \gamma(p)\|_\infty$$

Theorem 6.1 (Stability of Persistence Diagrams)

Let $f, g: X \to \mathbb{R}$ be two functions on a topological space X with $\|f - g\|_\infty \leq \varepsilon$. Then the bottleneck distance between their persistence diagrams satisfies $d_B(D(f), D(g)) \leq \varepsilon$. This stability theorem means that small perturbations of the function (e.g., small changes in network weights) produce small changes in the persistence diagram. Consequence for neural network analysis: persistence diagrams of activation functions or loss landscape sublevel sets vary continuously during training, enabling topological analysis of training dynamics that is robust to noise.

6.3 Topological Analysis of Loss Landscapes

The loss landscape $\mathscr{L}: \mathbb{R}^P \to \mathbb{R}$ can be analysed topologically via the sublevel set filtration $\{\mathscr{L} \leq c\}_{c \in \mathbb{R}}$. Connected components in H_0 correspond to disconnected minima basins; loops in H_1 correspond to "mountain passes" connecting basins. The **0-dimensional persistence** provides a rigorous measure of the number of distinct loss basins: a long-lived connected component (birth $b \approx \mathscr{L}_{min}$, death d large) is a deep, well-separated minimum; short-lived components are shallow local minima that merge quickly into deeper basins.

In overparameterised networks, topological analysis reveals that the loss landscape is **topologically trivial** at the global minimum: the set $\{\theta: \mathscr{L}(\theta) \leq \varepsilon\}$ for small ε is (approximately) connected, meaning that all global minima are accessible from each other via paths of near-zero loss. This is the **mode connectivity** phenomenon: linear or quadratic paths can connect different SGD solutions found from different random initialisations with near-zero loss along the entire path.

6.4 Topological Interpretation of Grokking

Grokking (delayed generalisation) can be characterised topologically. During the memorisation phase, the representation topology of training examples and

test examples are distinct: training examples form tight clusters (low H_0 persistence), while test examples are scattered. During the generalisation phase, the representation topology undergoes a phase transition: the representation space reorganises into a structure where training and test examples have similar topological signatures — the same connected components, the same loop structure.

Nanda et al.'s (2023) mechanistic account of grokking for modular arithmetic provides the precise topological picture: the **Fourier representation** $(\cos(2\pi ka/p), \sin(2\pi ka/p))$ corresponds to points on a circle in representation space. The H_1 persistent homology of the representation detects this circular structure: before grokking, H_1 is empty (no loops detected); after grokking, H_1 contains one long-lived loop corresponding to the circular Fourier structure. The grokking transition is precisely the moment this topological feature is born in the representation.

Concluding Remarks on Part VIII

The mathematical frameworks assembled in this part — POMDPs and multi-agent game theory, inference-time compute scaling, video diffusion, meta-learning, continual learning, and topological analysis — share a common theme: they address the problem of intelligence across time. Agentic systems must reason over extended horizons; meta-learning must leverage structure across a lifetime of tasks; continual learning must preserve knowledge across a stream of environments; topological analysis characterises the geometry of learning trajectories.

The connections to other parts of the treatise are numerous. GRPO and MAPPO are applications of the policy gradient theorem of Part V; EWC uses the Fisher information matrix of Part I; video flow matching applies the optimal transport framework of Part IV; the topological analysis of grokking connects to the algebraic topology of Part VI. Scientific AI (Part VII) increasingly incorporates meta-learning (few-shot adaptation to new protein families) and continual learning (updating models as new structural data arrives). The mathematical unity of the treatise reflects the mathematical unity of the discipline.

APPENDICES

Mathematical Notation · Theorems Reference · Proof Techniques · Index

Based on the work of Silvano Salvador

Appendix A: Mathematical Notation and Conventions

This appendix consolidates the notational conventions used throughout the eight parts of the treatise. The notation follows that of Salvador's Sezione 0, augmented as needed by standard conventions from the literature. Ambiguity is resolved in favour of clarity over brevity.

A.1 Sets, Spaces, and Structural Objects

Symbol	Meaning / Definition
$\mathbb{R}$, $\mathbb{C}$, $\mathbb{Z}$, $\mathbb{N}$	Real numbers, complex numbers, integers, natural numbers (0-inclusive)
$\mathbb{R}^n$, $\mathbb{R}^{m\times n}$	n-dimensional real vector space; space of real m×n matrices
$\mathbb{S}^{n-1}$	Unit (n-1)-sphere in $\mathbb{R}^n$: $\{x \in \mathbb{R}^n : \|x\|_2 = 1\}$
$\Delta(X)$	Set of probability distributions over X (probability simplex if X finite)
$\mathcal{H}$	Hilbert space; $\mathcal{G}$ denotes a general measurable space
$\mathcal{X}$, $\mathcal{Y}$	Input space, output space (generic)
$[n]$	The set {1, 2, ..., n}
$\mathcal{P}(X)$	Power set of X; $\mathcal{P}_k(X)$ = all k-element subsets
G, H	Groups (script letters); $g \in G$ denotes a group element
$GL(n)$, $O(n)$, $SO(n)$	General linear, orthogonal, special orthogonal groups over $\mathbb{R}$
$SE(3)$	Special Euclidean group in 3D: rigid motions (rotations + translations)
$\mathcal{G} = (V, E)$	Graph with vertex set V and edge set E

A.2 Vectors and Matrices

Symbol	Meaning
x, y, z	Column vectors (lower-case bold italic in equations)
x_i	i-th component of vector x; x_{ij} is (i,j) entry of matrix X
X, W, A, B	Matrices (upper-case); W^{(ℓ)} is the weight matrix at layer ℓ
X^{S}	Transpose of matrix X; X† (or X*) = conjugate transpose (Hermitian)
X^{-1}, X†	Matrix inverse; Moore-Penrose pseudoinverse
⊗	Kronecker product: (A ⊗ B)_{ij,kl} = A_{ik} B_{jl}
⊙	Hadamard (element-wise) product: (A ⊙ B)_{ij} = A_{ij} B_{ij}
diag(v)	Diagonal matrix with entries given by vector v
vec(X)	Column-major vectorisation of matrix X
Tr(X)	Trace of square matrix X = $\Sigma_i X_{ii}$
rank(X)	Rank (dimension of column space) of X
σ_i(X)	i-th singular value of X (σ_1 ≥ σ_2 ≥ ... ≥ 0)
‖X‖_F	Frobenius norm = (Σ_{ij} X^2_{ij})^{1/2} = (Σ_i σ_i^2)^{1/2}
‖X‖_*	Nuclear norm = Σ_i σ_i (sum of singular values)
$\|\mathbf{X}\|_2$	Spectral norm = σ_1(X) (largest singular value)
‖x‖_p	ℓ^p-norm = $(\Sigma_i \|x_i\|^p)^{1/p}$; ‖x‖_∞ = max_i $\|x_i\|$
⟨x, y⟩	Inner product = $x^T y$ (Euclidean); ⟨f, g⟩_$\mathcal{H}$ for Hilbert space inner product

A.3 Calculus and Optimisation

Symbol	Meaning
∂f/∂x	Partial derivative of scalar f with respect to x
∇_θ f(θ)	Gradient vector of f at θ: $(\partial f/\partial\theta_1, ..., \partial f/\partial\theta_p)^T$
∇^2_θ f(θ)	Hessian matrix of f: H_{ij} = $\partial^2 f / \partial\theta_i \partial\theta_j$
J_f(x)	Jacobian of vector-valued map f: (J_f)_{ij} = ∂f_i/∂x_j

Symbol	Meaning
δL/δf	Functional derivative of functional L with respect to function f
∂_θ L	Shorthand for ∇_θ L (log-derivative notation)
d/dt, $\dot{\mathbf{x}}$	Time derivative of x(t)
∘	Function composition: (f ∘ g)(x) = f(g(x))
f*	Convex conjugate (Legendre-Fenchel transform): f*(y) = sup_x {⟨x,y⟩ - f(x)}
prox_{λf}(x)	Proximal operator: arg min_z {f(z) + (1/2λ)‖z - x‖²}
∂f(x)	Subdifferential of convex f at x (set of subgradients)
O(·), Θ(·), Ω(·)	Landau asymptotic notation (upper, tight, lower bounds)

A.4 Probability and Statistics

Symbol	Meaning
p(x), q(x)	Probability density or mass function (context determines which)
P(A)	Probability of event A
$\mathbb{E}$_p[f(X)]	Expectation of f(X) under distribution p; $\mathbb{E}$[·] when p is clear from context
Var_p[X]	Variance = $\mathbb{E}$[(X - $\mathbb{E}$X)²]
Cov(X,Y)	Covariance = $\mathbb{E}$[(X - $\mathbb{E}$X)(Y - $\mathbb{E}$Y)^T]
$\mathcal{N}$(μ, Σ)	Multivariate Gaussian with mean μ ∈ $\mathbb{R}^n$ and covariance Σ ∈ $\mathbb{R}$^{n×n}
$\mathcal{N}$(0, I_d)	Standard d-dimensional Gaussian
Cat(π)	Categorical distribution with probability vector π ∈ Δ^K
Ber(p)	Bernoulli distribution with success probability p
σ(x)	Logistic sigmoid: σ(x) = 1/(1 + e^{-x})
softmax(z)_i	Softmax: exp(z_i) / Σ_j exp(z_j)

Symbol	Meaning
lse(β, z)	Log-sum-exp: (1/β) log Σᵢ exp(β z_i)
D_KL(p ‖ q)	KL divergence: 𝔼_p[log p(X)/q(X)] ≥ 0; equals 0 iff p = q a.e.
H(p)	Entropy: -𝔼_p[log p(X)]; H(p,q) = -𝔼_p[log q(X)] = cross-entropy
I(X;Y)	Mutual information: H(X) - H(X∣Y) = D_KL(p(x,y) ‖ p(x)p(y))
W_p(μ, ν)	p-Wasserstein distance between probability measures μ, ν
D_TV(p, q)	Total variation distance: (1/2) Σ_x ∣p(x) - q(x)∣
x ~ p	'x is distributed according to p' / 'x is sampled from p'
x ⊥ y ∣ z	Conditional independence of x and y given z

A.5 Deep Learning–Specific Notation

Symbol	Meaning
L(θ), ℒ(θ)	Loss function (L) and ELBO / variational objective ($\mathscr{L}$)
f_θ	Neural network with parameters θ; f_θ: $\mathscr{X} \to \mathscr{Y}$
a^{(ℓ)}, z^{(ℓ)}	Post-activation and pre-activation at layer ℓ
δ^{(ℓ)}	Error signal at layer ℓ during backpropagation
W^{(ℓ)}, b^{(ℓ)}	Weight matrix and bias vector at layer ℓ
σ(·)	Activation function (context: logistic sigmoid or generic nonlinearity)
ReLU(x)	Rectified linear unit: max(0, x); component-wise for vectors
GELU(x)	Gaussian error linear unit: x · Φ(x) where Φ is Gaussian CDF
Q, K, V	Query, Key, Value matrices in attention mechanism
Attn(Q,K,V)	Scaled dot-product attention: softmax($QK^T/\sqrt{d}$) · V
G(θ)	Fisher Information Matrix (FIM): 𝔼[∇log p · ∇log p^T]
Θ(x,x')	Neural Tangent Kernel (NTK) evaluated at inputs x, x'
π_θ(a∣s)	Policy: probability of action a in state s with parameters θ

Symbol	Meaning
$V^\pi(s)$, $Q^\pi(s,a)$	Value function and action-value function under policy π
$\mathcal{T}^\pi$, $\mathcal{T}^*$	Bellman evaluation and optimality operators
r_t, G_t	Reward at time t; return (discounted sum of rewards from t)
$\mathcal{D}$	Dataset or replay buffer
ε_t	Reparametrisation noise $\varepsilon \sim \mathcal{N}(0,I)$ in VAE / diffusion
$s_\theta(x,t)$	Score network approximating $\nabla_x \log p_t(x)$

Appendix B: Compendium of Key Theorems

This appendix provides a self-contained statement of the principal theorems referenced across the eight parts of the treatise, organised by mathematical domain. For each theorem, the originating chapter and key references are given. Proofs are sketched where the technique is instructive; complete proofs are found in the cited sources.

B.1 Linear Algebra and Matrix Theory

Theorem B.1 (Singular Value Decomposition)

Every matrix $A \in \mathbb{R}^{m\times n}$ admits the factorisation $A = U\Sigma V^T$ where $U \in O(m)$, $V \in O(n)$, and $\Sigma = \mathrm{diag}(\sigma_1,\ldots,\sigma_r, 0,\ldots,0)$ with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, $r = \mathrm{rank}(A)$.

The SVD is unique (up to sign) when the singular values are distinct.

Best rank-k approximation (Eckart-Young): $\|A - \hat{A}_k\|_F$ is minimised by $\hat{A}_k = U_k \Sigma_k V_k^T$ (first k columns of U, k×k top-left of Σ, first k columns of V). The error is $\|A - \hat{A}_k\|_F = (\Sigma_{i>k} \sigma_i^2)^{1/2}$.

Theorem B.2 (Marchenko-Pastur Law)

Let $X \in \mathbb{R}^{n\times p}$ have i.i.d. entries with mean 0 and variance σ^2/n; let $\gamma = p/n \to c \in (0,\infty)$.

The empirical spectral distribution of X^TX converges weakly a.s. to the Marchenko-Pastur law:

$$dF_c(\lambda) = \sqrt{(b-\lambda)(\lambda-a)} / (2\pi c\sigma^2\lambda) \cdot 1_{[a,b]}(\lambda) + \max(0, 1-1/c) \cdot \delta_0$$

where $a = \sigma^2(1 - \sqrt{c})^2$, $b = \sigma^2(1 + \sqrt{c})^2$.

Deviations from MP (heavy-tailed bulk, outlier eigenvalues) signal learned structure in trained networks.

Theorem B.3 (Weyl's Perturbation Theorem)

Let $A, E \in \mathbb{R}^{n\times n}$ be symmetric. Denote eigenvalues $\lambda_1(\cdot) \geq \ldots \geq \lambda_n(\cdot)$. Then:

$$\max_i |\lambda_i(A + E) - \lambda_i(A)| \leq \|E\|_2$$

Application: The Hessian $\nabla^2 L(\theta)$ of the loss is perturbed by mini-batch noise; the sharpness (largest Hessian eigenvalue) can change by at most the spectral norm of the noise. This bounds the rate of sharpness increase during training — a tool for analysing flat minimum methods (SAM).

B.2 Information Theory and Information Geometry

Theorem B.4 (Data Processing Inequality)

For any Markov chain $X \to Y \to Z$ (Y is a function of X, Z is a function of Y):

$I(X; Z) \le I(X; Y)$ and $I(X; Z) \le I(Y; Z)$

Mutual information cannot increase through processing. Consequence: no representation Z of input X can contain more information about label Y than X itself — the theoretical ceiling for learning is $I(X;Y)$.

Theorem B.5 (Fisher-Cramér-Rao Bound)

For any unbiased estimator $T(X_1,...,X_n)$ of a parameter $\theta \in \mathbb{R}^d$ from i.i.d. observations:

$\mathrm{Cov}(T) \ge G(\theta)^{-1} / n$ (in the Loewner / positive semidefinite sense)

where $G(\theta)$ is the Fisher Information Matrix. The bound is achieved by the MLE in regular families. Role in ML: the FIM is the metric on the statistical manifold; natural gradient descent is coordinate-invariant steepest descent in the FIM metric, not the Euclidean metric.

Theorem B.6 (Gibbs' Variational Principle)

For any distribution q and any measurable f:

$\log \mathbb{E}_p[\exp(f)] = \sup_q \{ \mathbb{E}_q[f] - D_{KL}(q \| p) \}$

The supremum is attained at $q^*(x) \propto p(x) \exp(f(x))$.

Consequences: (1) The ELBO identity in VAEs; (2) the optimal RLHF policy ($\pi^* \propto \pi_{ref} \cdot \exp(r/\beta)$); (3) the variational representation of the log-partition function in energy-based models.

B.3 Statistical Learning Theory

Theorem B.7 (PAC Learning Generalisation Bound via Rademacher Complexity)

Let $\mathscr{H}$ be a hypothesis class, $\mathscr{S} = \{(x_i, y_i)\}_{i=1}^m$ i.i.d. from D. With probability $\ge 1-\delta$:

for all $h \in \mathscr{H}$: $L_D(h) \le \hat{L}_S(h) + 2\mathfrak{R}_m(\mathscr{H}) + \sqrt{(\log(2/\delta) / 2m)}$

where $\mathbb{E}_m(\in) = _i _\sigma[\sup_{\{h_i\}} (1/m) \Sigma_i \sigma_i h(x_i)]$ is the empirical Rademacher complexity.

For neural networks with bounded weight norms, $\mathfrak{R}_m \le C \cdot \|W\|_F / \sqrt{m}$ (matrix spectral norms).

Theorem B.8 (Neural Tangent Kernel Convergence)

Consider a fully-connected network f_θ: $\mathbb{R}^{n_0} \to \mathbb{R}$ with widths $n_1 = \ldots = n_L = n \to \infty$.
The NTK $\Theta_n(x,x') = \Sigma_p (\partial f_\theta(x)/\partial\theta_p)(\partial f_\theta(x')/\partial\theta_p)$ converges in probability to a deterministic kernel $\Theta\infty(x,x')$ as $n \to \infty$. Under gradient flow on the squared loss, if the NTK Gram matrix $K = (\Theta\infty(x_i,x_j))$
has minimum eigenvalue $\lambda_{min} > 0$, the training loss converges exponentially:
$L(t) \le L(0) \cdot \exp(-2\lambda_{min} t)$

B.4 Optimisation

Theorem B.9 (Convergence of SGD for Non-convex Functions)

Let f: $\mathbb{R}^d \to \mathbb{R}$ be L-smooth ($\|\nabla f(x) - \nabla f(y)\| \le L\|x-y\|$) and bounded below.
SGD with step size $\eta_t = \eta/\sqrt{t}$, where gradient estimates satisfy $\mathbb{E}[\|\hat{g}_t - \nabla f(\theta_t)\|^2] \le \sigma^2$:
$(1/T)\,\Sigma_{t=1}^T\, \mathbb{E}[\|\nabla f(\theta_t)\|^2] \le O(\sigma/\sqrt{T}) + O(L\,\eta\,\sigma^2)$
So after $T = O(\varepsilon^{-4})$ steps, expected gradient norm is $\le \varepsilon$ (first-order stationarity).
Adam achieves the same rate with per-coordinate learning rates, often with better constants.

Theorem B.10 (Mirror Descent and Natural Gradient as Bregman Projections)

Natural gradient descent is equivalent to Mirror Descent with the Bregman divergence
$D_F(\theta, \theta') = F(\theta) - F(\theta') - \langle \nabla F(\theta'), \theta - \theta' \rangle$ where F is the log-partition function:
$\theta_{t+1} = \arg\min_\theta \{ \eta \langle \nabla L(\theta_t), \theta \rangle + D_{KL}(p_\theta \,\|\, p_{\theta_t}) \}$
This makes explicit that natural gradient descent follows the steepest descent path in the Riemannian manifold of distributions, not in parameter space.

B.5 Generative Models

Theorem B.11 (ELBO Decomposition for VAEs)

For any distribution $q_\varphi(z|x)$ (approximate posterior) and model $p_\theta(x,z) = p(z)p_\theta(x|z)$:
$\log p_\theta(x) = \mathscr{L}(\theta,\varphi;x) + D_{KL}(q_\varphi(z|x) \,\|\, p_\theta(z|x))$

where $\mathscr{L} = \mathbb{E}_{q_\varphi}[\log p_\theta(x|z)] - D_{KL}(q_\varphi(z|x) \| p(z))$ is the ELBO.
Since $D_{KL} \geq 0$, the ELBO is a lower bound: $\log p_\theta(x) \geq \mathscr{L}(\theta,\varphi;x)$.
Maximising $\mathscr{L}$ jointly over θ, φ simultaneously maximises the log-likelihood lower bound and minimises the approximation gap between q_φ and the true posterior.

Theorem B.12 (Anderson Reverse-Time SDE)

Let $\{x_t\}_{t\in[0,T]}$ satisfy the forward SDE: $dx_t = f(x_t,t)dt + g(t)dW_t$
with marginal densities $p_t(x)$. The exact reverse-time SDE (Anderson 1982) is:

$$dx_t = [f(x_t,t) - g(t)^2 \nabla_x \log p_t(x_t)]\, dt + g(t)\, d\bar{W}_t$$

where $\bar{W}_t$ is a reverse-time Brownian motion.
The score function $\nabla_x \log p_t$ can be approximated by a neural network $s_\theta(x,t)$ trained via denoising score matching, converting learning the reverse SDE to supervised regression.

Theorem B.13 (Wasserstein GAN Duality)

By the Kantorovich-Rubinstein theorem, the 1-Wasserstein distance between p_{data} and p_θ is:

$$W_1(p_{data}, p_\theta) = \sup_{f:\, \mathrm{Lip}(f) \leq 1} \{ \mathbb{E}_{x\sim p_{data}}[f(x)] - \mathbb{E}_{x\sim p_\theta}[f(x)] \}$$

Wasserstein GANs optimise this by parameterising the 1-Lipschitz critic f with a neural network (weight clipping or gradient penalty to enforce the Lipschitz constraint), and training the generator to minimise W_1 rather than JS divergence. The advantage: W_1 is continuous even when supports are disjoint.

B.6 Reinforcement Learning

Theorem B.14 (Bellman Contraction and Value Iteration Convergence)

The Bellman optimality operator $\mathscr{T}^*: \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ is a γ-contraction in $\|\cdot\|_\infty$:

$$\|\mathscr{T}^*V - \mathscr{T}^*U\|_\infty \leq \gamma \, \|V - U\|_\infty \quad \text{for all } V, U \in \mathbb{R}^{\mathcal{S}}$$

By Banach's fixed-point theorem, $V^* = \lim_{k\to\infty} (\mathscr{T}^*)^k V_0$ for any V_0, with

$$\|(\mathscr{T}^*)^k V_0 - V^*\|_\infty \leq \gamma^k \, \|V_0 - V^*\|_\infty$$

Policy Iteration converges in finitely many steps (at most $|\mathcal{A}|^{|\mathcal{S}|}$ policies).

Theorem B.15 (DPO Equivalence to RLHF, Rafailov et al. 2023)

Under the Bradley-Terry preference model $p(y^+ \succ y^- | x) = \sigma(r^*(x,y^+) - r^*(x,y^-))$,
the policy that minimises $L_{DPO}(\theta) = -\mathbb{E}[\log \sigma(\beta \log \pi_\theta(y^+|x)/\pi_{ref}(y^+|x) - \beta \log \pi_\theta(y^-|x)/\pi_{ref}(y^-|x))]$
is identical to the optimal policy of the KL-constrained RL problem:
$\pi^* = \arg\max_\pi \mathbb{E}[r^*(x,y)] - \beta D_{KL}(\pi \| \pi_{ref})$
DPO thus implicitly fits a reward model and optimises the RL objective simultaneously, without requiring an explicit reward model or RL training loop.

B.7 Algebraic Topology and Statistical Physics

Theorem B.16 (Persistent Homology Stability)

Let $f, g: \mathcal{K} \to \mathbb{R}$ be functions on a simplicial complex $\mathcal{K}$. The bottleneck distance between their persistence diagrams satisfies:
$d_B(PD(f), PD(g)) \leq \|f - g\|_\infty$
Consequence: persistent homology is a stable topological summary. Small perturbations in the loss function cannot produce large changes in the topology of its sublevel filtration. This makes persistent homology a robust tool for comparing loss landscapes across training runs.

Theorem B.17 (Hopfield Network Capacity: Classical vs Modern)

Classical Hopfield (binary): $M_{max} = 0.138\, N$ random patterns retrievable
Modern Hopfield (continuous): $M_{max} = \exp(\alpha N)$ patterns for some $\alpha > 0$
The exponential increase stems from replacing the bilinear energy with a log-sum-exp energy, which is equivalent to computing attention scores. The update rule becomes $\sigma_{new} = \Xi\, \text{softmax}(\beta\, \Xi^T \sigma)$, identifying Modern Hopfield retrieval with Transformer attention.

Appendix C: Core Proof Techniques

This appendix catalogues the principal proof techniques that recur throughout the treatise, providing brief explanations and examples of their use. Mastery of these techniques is prerequisite to reading the research literature.

C.1 The Log-Derivative (Score Function) Trick

Whenever an expectation $\mathbb{E}_{p_\theta}[f(x)]$ must be differentiated with respect to θ, the log-derivative identity $\nabla_\theta p_\theta(x) = p_\theta(x) \nabla_\theta \log p_\theta(x)$ converts the derivative of the integral into an integral:

$$\nabla_\theta \mathbb{E}_{p_\theta}[f(x)] = \mathbb{E}_{p_\theta}[f(x) \nabla_\theta \log p_\theta(x)]$$

This enables Monte Carlo gradient estimation by sampling $x \sim p_\theta$. Uses: REINFORCE, variational inference BBVI, policy gradient theorem. The resulting estimator is unbiased but high-variance; control variates (baselines) reduce variance without introducing bias.

C.2 The Reparametrisation Trick

For continuous distributions where $x = g_\theta(\varepsilon)$ with $\varepsilon \sim p(\varepsilon)$ independent of θ (e.g., $x = \mu_\theta + \sigma_\theta \varepsilon$, $\varepsilon \sim \mathcal{N}(0,I)$), the gradient passes through the sampling operation:

$$\nabla_\theta \mathbb{E}_{p_\theta(x)}[f(x)] = \mathbb{E}_{p(\varepsilon)}[\nabla_\theta f(g_\theta(\varepsilon))]$$

The reparametrisation trick decouples the stochastic node from the parameter θ, enabling low-variance gradients. Uses: VAE, SAC, normalising flows, continuous diffusion. Fails for discrete distributions — the Gumbel-Softmax relaxation provides a biased but low-variance alternative.

C.3 McDiarmid's Inequality and Uniform Convergence

McDiarmid's inequality bounds the deviation of a function of independent random variables:

$$\text{If } f(X_1,\ldots,X_m) \text{ satisfies: } \sup_{x_i,x'_i} |f(\ldots,x_i,\ldots) - f(\ldots,x'_i,\ldots)| \le c_i$$

$$\text{then: } P(f(X) - \mathbb{E}[f(X)] \ge t) \le \exp(-2t^2 / \Sigma_i c_i^2)$$

Uses: all PAC learning generalisation bounds. The function f is the empirical risk; the bounded difference condition holds with $c_i = 2/m$ for bounded losses. Combined with a union bound over $\mathcal{H}$ (or chaining for infinite $\mathcal{H}$), this yields uniform convergence bounds.

C.4 The Replica Method (Statistical Physics Heuristic)

The replica method computes quenched averages $\mathbb{E}_J[\log Z(J)]$ via the identity $\log Z = \lim_{n\to 0} \partial/\partial n\, \mathbb{E}[Z^n]$. For integer n, $Z^n = \Pi_{a=1}^{n} Z$ creates n independent copies ('replicas') that become coupled through the disorder average. After computing $\mathbb{E}[Z^n]$ analytically for integer n and identifying order parameters Q_{ab}, one analytically continues to $n \to 0$.

The method is non-rigorous (the $n \to 0$ limit lacks a general justification) but produces exact results that have subsequently been rigorously verified. Uses: Hopfield capacity, SK spin glass free energy, learning curve of linear classifiers, high-dimensional regression.

C.5 Stochastic Approximation and the ODE Method

Many RL and ML algorithms are stochastic approximation iterations: $\theta_{t+1} = \theta_t + \alpha_t\, h(\theta_t, \xi_t)$ where ξ_t is noise and h is the update direction. Convergence is established by showing that the 'mean ODE' $d\theta/dt = H(\theta) := \mathbb{E}[h(\theta, \xi)]$ has a globally attracting fixed point, and the stochastic iterates shadow this ODE (Ljung 1977, Borkar & Meyn 2000).

Key conditions: (1) Robbins-Monro step sizes $\Sigma\, \alpha_t = \infty$, $\Sigma\, \alpha_t^2 < \infty$; (2) h is bounded on compact sets; (3) the mean ODE is Lyapunov stable. Uses: Q-learning convergence (Watkins-Dayan), TD(λ) convergence, stochastic gradient methods.

Appendix D: Taxonomy of Neural Architectures and Inductive Biases

The following table organises major architectural families by their symmetry group (inductive bias), representative models, primary domain, and references to relevant chapters of the treatise.

Architecture Family	Symmetry / Inductive Bias	Representative Models	Domain	Treatise Part
MLP	None (dense)	FFN, MLP-Mixer	General	I
CNN	Translation equivariance	ResNet, EfficientNet, ConvNeXt	Vision	I, III
Transformer	Permutation equivariance + positional	GPT, BERT, ViT, Swin	Language, Vision	I, II
GNN	Graph isomorphism	GCN, GAT, MPNN, GraphTransformer	Graphs	II
LSTM/RNN	Sequential order	LSTM, GRU, Mamba, S4	Sequences	II
Equivariant Net	SE(3)/E(3) equivariance	SE(3)-TFN, EGNN, NequIP	Molecular/3D	I, II
VAE	Disentangled latent structure	β-VAE, NVAE, HiVAE	Generative	IV
GAN	Adversarial training	StyleGAN, BigGAN, CycleGAN	Generative	IV
NF	Invertibility + tractable Jacobian	RealNVP, Glow, NSF	Generative	IV
Diffusion	Reverse SDE	DDPM, LDM, DiT, SiT	Generative	IV
EBM	Energy function	EBM, JEM, Modern Hopfield	Generative/AM	IV, VI
RL Policy	Markov, temporal structure	PPO, SAC, TD3, AlphaZero	Control	V

Appendix E: Glossary of Technical Terms

A concise glossary of technical terms used throughout the treatise, with brief formal definitions.

Attention. A weighted sum of values, where weights are computed as a softmax of dot products between queries and keys. The core operation of Transformer models.

Backpropagation. Efficient computation of gradients in a computational graph via the chain rule, traversed in reverse topological order (backward mode autodiff).

Bayesian Inference. Updating a prior distribution $p(\theta)$ to a posterior $p(\theta \mid \text{data}) \propto p(\text{data} \mid \theta)p(\theta)$ using Bayes' theorem; used in BNNs and Laplace approximation.

Contrastive Loss. A loss that pushes together representations of matched pairs and pushes apart unmatched pairs; InfoNCE is the mutual-information-maximising variant.

Diffusion Model. A generative model that learns to reverse a Gaussian noising process via score matching or denoising score matching.

ELBO. Evidence Lower BOund: a tractable lower bound on $\log p(x)$ used in VAE and variational inference; ELBO = $\mathbb{E}_q[\log p(x \mid z)] - D_{KL}(q \| p)$.

Emergent Capability. A capability absent in small models and present in large models, appearing discontinuously as scale increases.

Equivariance. A map f is equivariant to group G if $f(gx) = gf(x)$ for all $g \in G$. Invariance is the special case $f(gx) = f(x)$.

Fisher Information Matrix. $G(\theta) = \mathbb{E}[\nabla \log p \cdot \nabla \log p^{T}]$; the Riemannian metric on the statistical manifold; curvature of the KL divergence to second order.

Flow Matching. A generative model that learns a vector field $v_\theta(x,t)$ whose ODE $dx/dt = v_\theta$ transports a source distribution to the data distribution.

Grokking. The phenomenon where a network memorises training data but later (after extended training) discovers a generalising algorithm.

Hilbert Space. A complete inner product space; RKHS is the Hilbert space of functions associated to a kernel, central to kernel methods and SVMs.

Importance Sampling. Estimating $\mathbb{E}_p[f]$ by sampling from q: $\mathbb{E}_p[f] = \mathbb{E}_q[f \cdot p/q]$. Used in off-policy RL (importance ratio π/π_{old}).

Inductive Bias. A constraint built into model architecture that makes certain hypotheses more likely, typically reflecting domain knowledge.

KL Divergence. $D_{KL}(p \| q) = \mathbb{E}_p[\log(p/q)] \geq 0$; measures 'extra bits' needed to encode p using code optimised for q; asymmetric.

LoRA. Low-Rank Adaptation: fine-tuning by learning $\Delta W = BA$ with $r \ll \min(m,n)$; reduces trainable parameters from mn to $r(m+n)$.

MCMC. Markov Chain Monte Carlo: algorithms (MH, HMC, NUTS) for sampling from unnormalised distributions by constructing ergodic Markov chains.

Mutual Information. $I(X;Y) = H(X) - H(X \mid Y) = D_{KL}(p(x,y) \| p(x)p(y))$; measures statistical dependence; central to information bottleneck and contrastive learning.

Natural Gradient. $\nabla^{nat}_\theta L = G(\theta)^{-1} \nabla_\theta L$; steepest descent in the Fisher-Rao metric; invariant under reparametrisation of θ.

NTK. Neural Tangent Kernel: the limiting kernel of an infinite-width network, constant during gradient flow training (lazy training regime).

Optimal Transport. $W_p(\mu,\nu)$: the cost of the cheapest way to transport mass from μ to ν; the p-Wasserstein distance; used in GANs, flow matching, and alignment.

Persistent Homology. Topological summary of a filtration of simplicial complexes; records birth and death of connected components, loops, and voids as scale varies.

Rademacher Complexity. $\mathfrak{R}_m(\mathscr{H})$: a data-dependent measure of hypothesis class capacity that tightly bounds generalisation; based on correlation with random sign patterns.

Replica Method. A statistical physics heuristic for computing $\mathbb{E}[\log Z]$ via analytic continuation from integer n (replicas) to $n \to 0$.

Residual Connection. $x_{\ell+1} = x_\ell + F(x_\ell)$; additive skip connection that stabilises gradient flow by providing a gradient highway of slope 1.

Score Function. $\nabla_x \log p(x)$; the gradient of the log-density; learned by score-matching and used to drive the reverse diffusion SDE.

Superposition. The hypothesis that neural networks represent more features than neurons by storing them as near-orthogonal directions in activation space.

VC Dimension. The largest set of points that can be shattered (all 2^n labellings achieved) by some hypothesis; measures expressiveness of a hypothesis class.

Wasserstein Distance. $W_p(\mu,\nu) = (\inf_{\gamma \in \Gamma(\mu,\nu)} \int c^p \, d\gamma)^{1/p}$; a proper metric on probability distributions that metrises weak convergence.

BIBLIOGRAPHY

Based on the work of Silvano Salvador

Bibliographic Note

This bibliography provides the complete references for claims, theorems, and empirical results cited across all parts of the treatise. Entries are organised thematically, mirroring the structure of the treatise itself, with a final section for the foundational source volumes by Silvano Salvador. Within each section, entries are ordered roughly chronologically to convey the intellectual history of each subfield.

Citation tags use the first three letters of the first author's surname followed by a two-digit year (e.g., [Vap95] for Vapnik 1995). In case of collision, a letter suffix disambiguates (e.g., [Sch17a] vs [Sch17b]). Preprints on arXiv are cited by their arXiv identifier where journal publication was not available at the time of writing.

I. Foundations: Linear Algebra, Optimisation, and Automatic Differentiation

Linear Algebra and Matrix Theory

[GVL13] G.H. Golub, C.F. Van Loan. *Matrix Computations, 4th ed.*. Johns Hopkins University Press, 2013. Standard reference for numerical linear algebra, SVD, and eigendecompositions.

[Bha97] R. Bhatia. *Matrix Analysis*. Springer Graduate Texts in Mathematics, 1997. Comprehensive treatment of spectral theory, perturbation bounds, and matrix functions.

[Wey12] H. Weyl. *Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen*. Mathematische Annalen, 71(4), 1912. Classical perturbation bound for eigenvalues; modern statement in Bhatia (1997).

[Mar67] V.A. Marchenko, L.A. Pastur. *Distribution of eigenvalues for some sets of random matrices*. Mathematics of the USSR-Sbornik, 1(4), 1967. Foundational RMT result for empirical spectral distribution of sample covariance matrices.

[Mar18] C.H. Martin, M.W. Mahoney. *Implicit Self-Regularization in Deep Neural Networks*. arXiv:1810.01075, 2018. Heavy-tail self-regularisation hypothesis; power-law spectral analysis of trained networks.

[Voi85] D. Voiculescu. *Symmetries of some reduced free product C*-algebras*. Springer LNM 1132, 1985. Introduction to free probability theory; S-transform multiplicativity.

[And10] G.W. Anderson, A. Guionnet, O. Zeitouni. *An Introduction to Random Matrices*. Cambridge University Press, 2010. Standard graduate textbook on RMT, Wigner semicircle, MP law, free probability.

Backpropagation and Automatic Differentiation

[Rum86] D.E. Rumelhart, G.E. Hinton, R.J. Williams. *Learning representations by back-propagating errors*. Nature, 323, 1986. Classic statement of backpropagation for multilayer networks.

[Bay88] Y. Le Cun. *A theoretical framework for back-propagation*. Proceedings of the 1988 Connectionist Models Summer School, 1988. Automatic differentiation perspective on backpropagation.

[Bay15] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind. *Automatic Differentiation in Machine Learning: a Survey*. JMLR, 18(153), 2018. Comprehensive survey of forward and reverse mode AD, and their ML applications.

[Pas19] A. Paszke et al.. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. NeurIPS 2019, 2019. PyTorch autodiff engine; dynamic computation graph and eager execution.

Optimisation

[Kin15] D.P. Kingma, J. Ba. *Adam: A Method for Stochastic Optimization.* ICLR 2015, 2015. Adam optimizer; adaptive moment estimation; bias correction.

[Los17] I. Loshchilov, F. Hutter. *Decoupled Weight Decay Regularization.* ICLR 2019, 2017. AdamW; shows that standard L2 and Adam weight decay are not equivalent.

[Bot10] L. Bottou. *Large-Scale Machine Learning with Stochastic Gradient Descent.* COMPSTAT 2010, 2010. Theoretical foundations of SGD for large-scale ML; learning rate schedules.

[Nes83] Y. Nesterov. *A method for solving the convex programming problem with convergence rate O(1/k²).* Soviet Mathematics Doklady, 27(2), 1983. Nesterov momentum; accelerated gradient descent for convex objectives.

[Mar20] D. Masters, C. Luschi. *Revisiting Small Batch Training for Deep Neural Networks.* arXiv:1804.07612, 2018. Empirical study of batch size effects on training dynamics and generalisation.

[For18] P. Foret, A. Kleiner, H. Mobahi, B. Neyshabur. *Sharpness-Aware Minimization for Efficiently Improving Generalization.* ICLR 2021, 2021. SAM: seeking flat minima by minimising the worst-case loss in a parameter neighbourhood.

[Ama98] S. Amari. *Natural Gradient Works Efficiently in Learning.* Neural Computation, 10(2), 1998. Natural gradient descent; Fisher information metric; invariance under reparametrisation.

[Mar15] J. Martens, R. Grosse. *Optimizing Neural Networks with Kronecker-factored Approximate Curvature.* ICML 2015, 2015. K-FAC: Kronecker-factored approximation of the Fisher information matrix.

II. Architectures and Geometric Inductive Biases

Transformer Architecture

[Vas17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin. *Attention Is All You Need.* NeurIPS 2017, 2017. Original Transformer architecture; multi-head self-attention; encoder-decoder.

[Dev19] J. Devlin, M.W. Chang, K. Lee, K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* NAACL 2019, 2019. Masked language modelling pre-training; bidirectional attention.

[Bro20] T.B. Brown et al.. *Language Models are Few-Shot Learners.* NeurIPS 2020, 2020. GPT-3 (175B); few-shot and zero-shot in-context learning; scaling to 175B parameters.

[Dao22] T. Dao, D. Fu, S. Ermon, A. Rudra, C. Ré. *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness.* NeurIPS 2022, 2022. IO-aware tiled attention; reduces memory from $O(S^2)$ to $O(S)$; enables long contexts.

[Dao23] T. Dao. *FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning.* ICLR 2024, 2023. Improved work distribution and parallelism for modern GPU architectures.

[Su24] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, Y. Liu. *RoFormer: Enhanced Transformer with Rotary Position Embedding.* Neurocomputing, 2024. RoPE: rotary position embeddings; multiplicative position encoding compatible with linear attention.

[Kat23] A. Katharopoulos, A. Vyas, N. Pappas, F. Fleuret. *Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention.* ICML 2020, 2020. Linear attention via kernel feature maps; reduces attention complexity from $O(S^2)$ to $O(S)$.

[Mit23] D. De, S. Smith, A. Fernando et al.. *Griffin: Mixing Gated Linear Recurrences with Local Attention.* arXiv:2402.19427, 2024. Hybrid linear-recurrence plus local attention; efficient long-sequence modelling.

Equivariant and Geometric Architectures

[Coh16] T.S. Cohen, M. Welling. *Group Equivariant Convolutional Networks.* ICML 2016, 2016. G-CNNs; equivariance to discrete symmetry groups; group convolution theorem.

[Tho18] N. Thomas et al.. *Tensor Field Networks: Rotation- and Translation-Equivariant Neural Networks for 3D Point Clouds.* arXiv:1802.08219, 2018. SE(3)-equivariant networks via spherical harmonic tensor products.

[Jum21] J. Jumper et al.. *Highly accurate protein structure prediction with AlphaFold.* Nature, 596, 2021. AlphaFold 2; IPA mechanism; FAPE loss; SE(3)-equivariance in protein structure prediction.

[Bro22] J.L. Batzner et al.. *E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials.* Nature Communications, 13, 2022. NequIP; SE(3)-equivariant GNNs for molecular dynamics force fields.

Graph Neural Networks

[Kip17] T.N. Kipf, M. Welling. *Semi-Supervised Classification with Graph Convolutional Networks.* ICLR 2017, 2017. GCN; spectral graph convolution; Chebyshev polynomial approximation.

[Vel18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio. *Graph Attention Networks.* ICLR 2018, 2018. GAT; attention-weighted aggregation over graph neighbourhoods.

[Xu19] K. Xu, W. Hu, J. Leskovec, S. Jegelka. *How Powerful are Graph Neural Networks?.* ICLR 2019, 2019. WL-test equivalence of MPNNs; GIN architecture achieving maximum WL-expressiveness.

[Bod22] C. Bodnar, F. Di Giovanni, B. Chamberlain, P. Liò, M. Bronstein. *Neural Sheaf Diffusion: A Topological Perspective on Heterophily in GNNs.* NeurIPS 2022, 2022. Cellular sheaves on graphs; sheaf Laplacian; sheaf neural networks.

Sequential and State Space Models

[Hoc97] S. Hochreiter, J. Schmidhuber. *Long Short-Term Memory.* Neural Computation, 9(8), 1997. LSTM; gated cell state; gradient highways through forget gate.

[Gu22] A. Gu, K. Goel, C. Ré. *Efficiently Modeling Long Sequences with Structured State Spaces.* ICLR 2022, 2022. S4; HiPPO matrices; FFT-based SSM convolution; O(L log L) computation.

[Gu23] A. Gu, T. Dao. *Mamba: Linear-Time Sequence Modeling with Selective State Spaces.* arXiv:2312.00752, 2023. Selective SSM; content-dependent B,C matrices; hardware-aware parallel scan.

[Pen23] B. Peng et al.. *RWKV: Reinventing RNNs for the Transformer Era.* EMNLP 2023, 2023. RWKV; linear attention as recurrence; WKV operator.

[Pol23] M. Poli et al.. *Hyena Hierarchy: Towards Larger Convolutional Language Models.* ICML 2023, 2023. Hyena; subquadratic long convolutions; implicit parameterisation.

III. Statistical Learning Theory and Probabilistic Methods

Learning Theory

[Vap95] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995. VC dimension, SRM, SVM; PAC learning framework.

[Bar02] P.L. Bartlett, S. Mendelson. *Rademacher and Gaussian Complexities: Risk Bounds and Structural Results*. JMLR, 3, 2002. Rademacher complexity; data-dependent generalisation bounds; tighter than VC.

[Ney15] B. Neyshabur, R. Tomioka, N. Srebro. *Norm-Based Capacity Control in Neural Networks*. COLT 2015, 2015. Spectral norm bounds for neural networks; parameter norm vs capacity.

[Bel20] M. Belkin, D. Hsu, S. Ma, S. Mandal. *Reconciling modern machine-learning practice and the classical bias-variance tradeoff*. PNAS, 116(32), 2019. Double descent; benign overfitting; interpolation threshold.

[Jac18] A. Jacot, F. Gabriel, C. Hongler. *Neural Tangent Kernel: Convergence and Generalization in Neural Networks*. NeurIPS 2018, 2018. NTK; infinite-width convergence; lazy training regime.

[Aro19] S. Arora, S. Du, W. Hu, Z. Li, R. Wang. *Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Neural Networks*. ICML 2019, 2019. NTK convergence guarantees for two-layer networks; overparameterised regime.

Bayesian Methods and MCMC

[Nea93] R.M. Neal. *Probabilistic Inference Using Markov Chain Monte Carlo Methods*. Technical Report CRG-TR-93-1, University of Toronto, 1993. Foundational reference for HMC in ML; MCMC for neural networks.

[Hof14] M.D. Hoffman, A. Gelman. *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo*. JMLR, 15(1), 2014. NUTS; adaptive HMC; No-U-Turn criterion; basis for Stan's sampler.

[Wel11] M. Welling, Y.W. Teh. *Bayesian Learning via Stochastic Gradient Langevin Dynamics*. ICML 2011, 2011. SGLD; stochastic gradient MCMC; injecting gradient noise for posterior sampling.

[Mac03] D.J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. Comprehensive Bayesian ML reference; Laplace approximation; free energy.

Variational Inference and Uncertainty Quantification

[Kin14] D.P. Kingma, M. Welling. *Auto-Encoding Variational Bayes*. ICLR 2014, 2014. VAE; ELBO; reparametrisation trick; amortised variational inference.

[Res15] D.J. Rezende, S. Mohamed. *Variational Inference with Normalizing Flows*. ICML 2015, 2015. Normalising flows for flexible approximate posteriors; planar/radial flows.

[Bur16] Y. Burda, R. Grosse, R. Salakhutdinov. *Importance Weighted Autoencoders*. ICLR 2016, 2016. IWAE; tighter ELBO via importance-weighted samples; gap to log p(x).

[Ang19] A. Angelopoulos, S. Bates. *A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification*. arXiv:2107.07511, 2021. Conformal prediction; distribution-free coverage guarantees for any predictor.

[Lac17] B. Lakshminarayanan, A. Pritzel, C. Blundell. *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. NeurIPS 2017, 2017. Deep ensembles; calibrated uncertainty via ensemble disagreement.

IV. Generative Models

GANs

[Goo14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. *Generative Adversarial Networks.* NeurIPS 2014, 2014. Original GAN; minimax game; JS divergence minimisation.

[Arj17] M. Arjovsky, S. Chintala, L. Bottou. *Wasserstein Generative Adversarial Networks.* ICML 2017, 2017. WGAN; Kantorovich-Rubinstein duality; gradient penalty for Lipschitz constraint.

[Kar20] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, T. Aila. *Analyzing and Improving the Image Quality of StyleGAN.* CVPR 2020, 2020. StyleGAN2; style-based generator; path length regularisation; FID records.

Normalising Flows

[Die14] L. Dinh, D. Krueger, Y. Bengio. *NICE: Non-linear Independent Components Estimation.* ICLR Workshop 2015, 2014. NICE; additive coupling layers; first tractable normalising flow.

[Die17] L. Dinh, J. Sohl-Dickstein, S. Bengio. *Density estimation using Real-valued Non-Volume Preserving (Real NVP) transformations.* ICLR 2017, 2017. RealNVP; affine coupling layers; bijective generative model.

[Kin18] D.P. Kingma, P. Dhariwal. *Glow: Generative Flow with Invertible 1×1 Convolutions.* NeurIPS 2018, 2018. Glow; learned permutation via invertible 1×1 convolutions; high-quality image synthesis.

[Dur19] C. Durkan, A. Bekasov, I. Murray, G. Papamakarios. *Neural Spline Flows.* NeurIPS 2019, 2019. NSF; monotone rational-quadratic splines as coupling layers; expressive normalising flow.

Score-Based Models and Diffusion

[Son19] Y. Song, S. Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution.* NeurIPS 2019, 2019. Score matching with Langevin dynamics; annealed IS for sample quality.

[Ho20] J. Ho, A. Jain, P. Abbeel. *Denoising Diffusion Probabilistic Models.* NeurIPS 2020, 2020. DDPM; Gaussian diffusion; denoising score matching as variational lower bound.

[Son21] Y. Song, J. Sohl-Dickstein, D.P. Kingma, A. Kumar, S. Ermon, B. Poole. *Score-Based Generative Modeling through Stochastic Differential Equations.* ICLR 2021, 2021. Unification of SMLD and DDPM via SDEs; VE/VP/subVP SDEs; probability flow ODE.

[Son22] J. Song, C. Meng, S. Ermon. *Denoising Diffusion Implicit Models*. ICLR 2021, 2021. DDIM; deterministic sampling from diffusion models; 10–50× inference speedup.

[Lip22] W. Lipman, R. Chen, H. Ben-Hamu, M. Nickel, M. Le. *Flow Matching for Generative Modeling*. ICLR 2023, 2022. Flow Matching; conditional vector fields; OT-FM for straight flows.

[Rom22] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. CVPR 2022, 2022. LDM / Stable Diffusion; diffusion in VQ-regularised latent space; cross-attention conditioning.

[Son23] Y. Song, P. Dhariwal, M. Chen, I. Sutskever. *Consistency Models*. ICML 2023, 2023. Consistency models; self-consistency property; direct one-step generation.

Self-Supervised and Contrastive Learning

[Che20] T. Chen, S. Kornblith, M. Norouzi, G. Hinton. *A Simple Framework for Contrastive Learning of Visual Representations*. ICML 2020, 2020. SimCLR; NT-Xent loss; data augmentation for contrastive self-supervised learning.

[Gri20] J.B. Grill et al.. *Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning*. NeurIPS 2020, 2020. BYOL; self-distillation without negative pairs; online/target network asymmetry.

[Bar21] J. Zbontar, L. Jing, I. Misra, Y. LeCun, S. Deny. *Barlow Twins: Self-Supervised Learning via Redundancy Reduction*. ICML 2021, 2021. Barlow Twins; cross-correlation matrix towards identity; information redundancy.

Energy-Based Models and Hopfield

[Ram21] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, L. Gruber, M. Holzleitner, M. Pavlović, G.K. Sandve, V. Greiff, D. Kreil, M. Kopp, G. Klambauer, J. Brandstetter, S. Hochreiter. *Hopfield Networks is All You Need*. ICLR 2021, 2021. Modern Hopfield networks; exponential capacity; equivalence with attention.

[Lec06] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. Huang. *A Tutorial on Energy-Based Learning*. Predicting Structured Data, MIT Press, 2006. EBMs; energy function learning; contrastive divergence.

V. Reinforcement Learning and Alignment

MDP Theory and Value-Based Methods

[Put94] M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994. Standard MDP reference; existence and uniqueness of optimal policy; PI convergence.

[Wat92] C.J.C.H. Watkins, P. Dayan. *Q-learning*. Machine Learning, 8(3-4), 1992. Q-learning algorithm; convergence proof via stochastic approximation theory.

[Mni15] V. Mnih et al.. *Human-level control through deep reinforcement learning*. Nature, 518, 2015. DQN; experience replay; target network; Atari benchmark.

[Bel17] M.G. Bellemare, W. Dabney, R. Munos. *A Distributional Perspective on Reinforcement Learning*. ICML 2017, 2017. Distributional RL; C51; Wasserstein contraction; return distribution.

Policy Gradient Methods

[Wil92] R.J. Williams. *Simple statistical gradient-following algorithms for connectionist reinforcement learning*. Machine Learning, 8(3-4), 1992. REINFORCE; score function gradient estimator; baseline variance reduction.

[Sut00] R.S. Sutton, D.A. McAllester, S.P. Singh, Y. Mansour. *Policy Gradient Methods for Reinforcement Learning with Function Approximation*. NeurIPS 2000, 2000. Policy Gradient Theorem; actor-critic; compatible function approximation.

[Sch16] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. ICLR 2016, 2016. GAE; bias-variance tradeoff in advantage estimation; λ interpolation.

[Sch15] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz. *Trust Region Policy Optimization*. ICML 2015, 2015. TRPO; KL-constrained policy update; Fisher-vector products via conjugate gradient.

[Sch17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov. *Proximal Policy Optimization Algorithms*. arXiv:1707.06347, 2017. PPO; clipped importance ratio surrogate; simpler and more robust than TRPO.

MaxEnt RL and Continuous Control

[Haa18] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. ICML 2018, 2018. SAC; soft Bellman equations; entropy-regularised RL; automatic temperature tuning.

[Fuj18] S. Fujimoto, H. van Hoof, D. Meger. *Addressing Function Approximation Error in Actor-Critic Methods.* ICML 2018, 2018. TD3; clipped double Q; target policy smoothing; delayed policy updates.

[Lil16] T.P. Lillicrap et al.. *Continuous control with deep reinforcement learning.* ICLR 2016, 2016. DDPG; deterministic policy gradient; actor-critic for continuous action spaces.

Model-Based RL

[Jan19] M. Janner, J. Fu, M. Zhang, S. Levine. *When to Trust Your Model: Model-Based Policy Optimization.* NeurIPS 2019, 2019. MBPO; theoretical bound on model error; short model rollouts for safe data augmentation.

[Haf19] D. Hafner, T. Lillicrap, J. Ba, M. Norouzi. *Dream to Control: Learning Behaviors by Latent Imagination.* ICLR 2020, 2019. Dreamer; RSSM; latent imagination; policy optimisation via imagined trajectories.

[Haf23] D. Hafner, J. Lillicrap, P. Norouzi, T.P. Lillicrap. *Mastering Diverse Domains through World Models.* arXiv:2301.04104, 2023. DreamerV3; universal hyperparameters; symlog predictions; free-bits KL.

[Sil18] D. Silver et al.. *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.* Science, 362, 2018. AlphaZero; MCTS + neural network; self-play without human knowledge.

[Sch20] J. Schrittwieser et al.. *Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model.* Nature, 588, 2020. MuZero; latent dynamics model; model-based planning without observation reconstruction.

RLHF, Alignment, and Preference Learning

[Chr17] P.F. Christiano, J. Leike, T.B. Brown, M. Martic, S. Legg, D. Amodei. *Deep Reinforcement Learning from Human Preferences.* NeurIPS 2017, 2017. RLHF; preference learning from human comparisons; reward model training.

[Sti20] N. Stiennon et al.. *Learning to summarize from human feedback.* NeurIPS 2020, 2020. RLHF for summarisation; PPO + reward model; first large-scale NLP application.

[Ouy22] L. Ouyang et al.. *Training language models to follow instructions with human feedback.* NeurIPS 2022, 2022. InstructGPT; RLHF pipeline for instruction following; PPO-based fine-tuning.

[Raf23] R. Rafailov, A. Sharma, E. Mitchell, C.D. Manning, S. Ermon, C. Finn. *Direct Preference Optimization: Your Language Model is Secretly a Reward Model.* NeurIPS 2023, 2023. DPO; reward reparametrisation; elimination of explicit RL stage from RLHF.

[Eth24] K. Ethayarajh, W. Xu, N. Muennighoff, J. Jurafsky, D. Kiela. *KTO: Model Alignment as Prospect Theoretic Optimization.* ICML 2024, 2024. KTO; Kahneman-Tversky prospect theory loss; binary feedback without preference pairs.

[Men24] Y. Meng, M. Xia, D. Chen. *SimPO: Simple Preference Optimization with a Reference-Free Reward.* NeurIPS 2024, 2024. SimPO; length-normalised reward; margin-based loss; reference-free alignment.

[Bai22] Y. Bai et al.. *Constitutional AI: Harmlessness from AI Feedback.* arXiv:2212.08073, 2022. Constitutional AI; RLAIF; critique-revision self-improvement; scalable oversight.

[Gao22] L. Gao, J. Schulman, J. Hilton. *Scaling Laws for Reward Model Overoptimization.* arXiv:2210.10760, 2022. KL-return Pareto frontier; Goodhart's law in RLHF; reward hacking dynamics.

Emergent Capabilities and In-Context Learning

[Wei22] J. Wei et al.. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.* NeurIPS 2022, 2022. CoT prompting; intermediate reasoning steps; arithmetic, commonsense, symbolic tasks.

[Wei23] J. Wei, Y. Tay, R. Bommasani et al.. *Emergent Abilities of Large Language Models.* TMLR 2022, 2022. Operational definition of emergence; phase transitions in capability with scale.

[Xie22] S.M. Xie, A. Raghunathan, P. Liang, T. Ma. *An Explanation of In-context Learning as Implicit Bayesian Inference.* ICLR 2022, 2022. ICL as Bayesian inference over latent concepts; posterior concentration with examples.

[Aky22] E. Akyürek, D. Schuurmans, J. Andreas, T. Ma, D. Zhou. *What learning algorithm is in-context learning? Investigations with linear models.* ICLR 2023, 2022. ICL as gradient descent; Transformer layers implement one step of ridge regression.

[Fen23] N. Feng, Q. Zhou, C. Gu, H. He. *Towards Revealing the Mystery behind Chain of Thought: a Theoretical Perspective.* NeurIPS 2023, 2023. CoT and circuit complexity; TC^0 vs NC^1; depth-multiplying effect of chain of thought.

[Lew20] P. Lewis et al.. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.* NeurIPS 2020, 2020. RAG; dense retrieval + generation; open-domain QA.

Mechanistic Interpretability

[Ola20] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, S. Carter. *Zoom In: An Introduction to Circuits.* Distill, 2020. Circuits framework; curve detectors; multimodal neurons; circuit analysis in CNNs.

[Ols22] K. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma et al.. *In-context Learning and Induction Heads.* Transformer Circuits Thread, 2022. Induction heads; two-layer circuit; phase transition during training.

[Elh22] N. Elhage, T. Henighan, A. Joseph, N. Nanda, N. Olsson et al.. *Toy Models of Superposition*. Transformer Circuits Thread, 2022. Superposition hypothesis; polytope geometry of feature representation; phase transitions.

[Cun23] H. Cunningham, A. Ewart, L. Riggs, R. Huben, L. Sharkey. *Sparse Autoencoders Find Highly Interpretable Features in Language Models*. arXiv:2309.08600, 2023. SAEs for mechanistic interpretability; monosemantic feature discovery.

VI. Advanced Mathematical Structures

Statistical Physics of Learning

[Hop82] J.J. Hopfield. *Neural networks and physical systems with emergent collective computational abilities.* PNAS, 79(8), 1982. Classical Hopfield network; energy function; memory capacity and retrieval.

[Ama87] D.J. Amit, H. Gutfreund, H. Sompolinsky. *Statistical Mechanics of Neural Networks near Saturation.* Annals of Physics, 173(1), 1987. Replica calculation of Hopfield capacity; AGS critical load $\alpha_c \approx 0.138$.

[Mez87] M. Mézard, G. Parisi, M. Virasoro. *Spin Glass Theory and Beyond.* World Scientific, 1987. Replica method; Parisi RSB; SK model exact solution; spin glass theory.

[Eng01] A. Engel, C. Van den Broeck. *Statistical Mechanics of Learning.* Cambridge University Press, 2001. Statistical physics approach to ML; storage capacity; generalisation in perceptrons.

[Bor22] B. Bordelon, C. Pehlevan. *Spectral Bias Outside the Neural Tangent Kernel Regime.* ICLR 2022, 2022. Spectral learning curves; power-law scaling from data covariance spectrum.

Algebraic Topology for ML

[Ede08] H. Edelsbrunner, J. Harer. *Persistent Homology — a Survey.* Contemp. Mathematics 453, 2008. Persistent homology; filtrations; persistence diagrams; Betti numbers.

[Coh07] D. Cohen-Steiner, H. Edelsbrunner, J. Harer. *Stability of Persistence Diagrams.* Discrete & Computational Geometry, 37(1), 2007. Bottleneck stability theorem; robustness of persistent homology to perturbations.

[Car09] G. Carlsson. *Topology and Data.* Bulletin of the AMS, 46(2), 2009. TDA survey; mapper algorithm; applications to real data.

[Han20] J. Hansen, R. Ghrist. *Opinion Dynamics on Discourse Sheaves.* SIAM J. Applied Mathematics, 81(5), 2020. Cellular sheaves on graphs; sheaf Laplacian; connection Laplacian.

[Pow22] A. Power, Y. Gur-Ari, D. Bahri, R. Anderson, J. Steinhardt. *Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets.* arXiv:2201.02177, 2022. Grokking phenomenon; delayed generalisation; algorithmic datasets.

Causal Inference and OOD Generalisation

[Pea09] J. Pearl. *Causality: Models, Reasoning, and Inference, 2nd ed..* Cambridge University Press, 2009. SCM; do-calculus; interventional and counterfactual distributions.

[Pet17] J. Peters, D. Janzing, B. Schölkopf. *Elements of Causal Inference*. MIT Press, 2017. Causal models; identifiability; ICM principle; invariant causal prediction.

[Arj19] M. Arjovsky, L. Bottou, I. Gulrajani, D. Lopez-Paz. *Invariant Risk Minimization*. arXiv:1907.02893, 2019. IRM; invariant predictors across environments; causal feature selection.

[Sag20] S. Sagawa, P.W. Koh, T.B. Hashimoto, P. Liang. *Distributionally Robust Neural Networks for Group Shifts*. ICLR 2020, 2020. Group DRO; worst-group accuracy; underrepresented group robustness.

Scaling Laws and Emergence

[Kap20] J. Kaplan, S. McCandlish, T. Henighan et al.. *Scaling Laws for Neural Language Models*. arXiv:2001.08361, 2020. Power-law scaling in N, D, C; compute-optimal allocation; chinchilla precursor.

[Hof22] J. Hoffmann et al.. *Training Compute-Optimal Large Language Models*. NeurIPS 2022, 2022. Chinchilla scaling law; equal scaling of N and D; 20 tokens per parameter.

[Sch23] R. Schaeffer, B. Miranda, S. Koyejo. *Are Emergent Abilities of Large Language Models a Mirage?*. NeurIPS 2023, 2023. Emergence as metric artefact; smooth improvement under linear metrics.

Large-Scale Fine-Tuning and Distributed Training

[Hu21] E.J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen. *LoRA: Low-Rank Adaptation of Large Language Models*. ICLR 2022, 2021. LoRA; low-rank weight perturbation; efficient fine-tuning of LLMs.

[Det23] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer. *QLoRA: Efficient Finetuning of Quantized LLMs*. NeurIPS 2023, 2023. QLoRA; 4-bit NF4 quantisation; double quantisation; 65B on single GPU.

[Raj20] S. Rajbhandari, J. Rasley, O. Ruwase, Y. He. *ZeRO: Memory Optimizations Toward Training Trillion Parameter Models*. SC 2020, 2020. ZeRO stages 1/2/3; parameter, gradient, and optimizer state sharding.

[Zha23] Y. Zhao, A. Gu, R. Varma, L. Luo et al.. *PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel*. VLDB 2023, 2023. FSDP; ZeRO-3 in PyTorch; per-unit parameter sharding; forward/backward gather.

[Che16] T. Chen, B. Xu, C. Zhang, C. Guestrin. *Training Deep Nets with Sublinear Memory Cost*. arXiv:1604.06174, 2016. Gradient checkpointing; √L memory-compute tradeoff; activation recomputation.

Multimodal Models

[Rad21] A. Radford, J.W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever. *Learning Transferable Visual Models from Natural Language Supervision*. ICML 2021, 2021. CLIP; contrastive image-text pre-training; zero-shot transfer; InfoNCE loss.

[Ala22] J.B. Alayrac et al.. *Flamingo: a Visual Language Model for Few-Shot Learning*. NeurIPS 2022, 2022. Flamingo; Perceiver Resampler; cross-attention VLM; few-shot multimodal learning.

[Liu23] H. Liu, C. Li, Q. Wu, Y.J. Lee. *Visual Instruction Tuning*. NeurIPS 2023, 2023. LLaVA; linear projector from CLIP to LLM; multimodal instruction tuning.

[Cut13] M. Cuturi. *Sinkhorn Distances: Lightspeed Computation of Optimal Transport Distances*. NeurIPS 2013, 2013. Entropic regularised OT; Sinkhorn algorithm; efficient W distance computation.

VII. Source Volumes: The Salvador Cycle

This treatise is a synthesis and commentary on the following eleven volumes by Silvano Salvador, which constitute the primary source material. Each volume is identified by its title and the sections it covers from the Salvador numbering system.

Volume	Title	Coverage	Topics
Matematica delle reti neurali (Volume 1)	Linear Algebra for Neural Networks	Sections §1–§90	Vectors, matrices, eigendecompositions, SVD, dimensionality reduction.
Matematica delle reti neurali (Volume 2)	Calculus, Backpropagation, and Optimisation	Sections §91–§200	Gradient descent, backpropagation, Adam, loss landscapes, normalisation.
Matematica delle reti neurali (Volume 3)	Statistical Learning Theory	Sections §201–§280	VC dimension, Rademacher complexity, PAC learning, NTK, double descent.
Matematica delle reti neurali (Volume 4)	Information Geometry	Sections §281–§360	Fisher metric, natural gradient, K-FAC, information bottleneck, RKHS.
Matematica delle reti neurali (Volume 5)	Architectures and Equivariance	Sections §361–§450	Transformer, attention variants, CNN equivariance, GNN, WL-test.
Matematica delle reti neurali (Volume 6)	Generative Models I	Sections §451–§530	VAE, GAN, normalising flows, score matching, DDPM, DDIM.
Matematica delle reti neurali (Volume 7)	Generative Models II and Diffusion	Sections §531–§600	SDE diffusion, consistency models, flow matching, LDM, EBM, Hopfield.

Volume	Title	Coverage	Topics
Matematica delle reti neurali (Volume 8)	Reinforcement Learning	Sections §601–§660	MDP, Bellman, policy gradients, SAC, TD3, MBRL, RLHF, DPO.
Matematica delle reti neurali (Volume 9)	Bayesian and Probabilistic Methods	Sections §661–§720	MCMC, HMC, NUTS, SGLD, VI, ELBO, BBVI, IWAE, BNN, conformal prediction.
Matematica delle reti neurali (Volume 10)	Advanced Mathematics	Sections §721–§790	Spin glasses, replicas, free probability, algebraic topology, causal inference.
Matematica delle reti neurali (Volume 11)	Scaling, Fine-Tuning, and Multimodal	Sections §791–§860	Scaling laws, emergence, LoRA, QLoRA, ZeRO, FSDP, CLIP, VLMs.

The section numbering §1–§860 is the unified cross-volume numbering used throughout the present treatise to pinpoint specific results within the Salvador cycle.

Appendix B: Technical Supplement for Part VII — Scientific AI and Geometric Deep Learning

This appendix collects detailed proofs, derivations, and reference tables that support the main text of Part VII. All notation follows Appendix A. Equation numbers of the form (B.x) refer exclusively to this appendix.

B.1 Detailed Proof of IPA SE(3) Invariance (Theorem 2.2 of Part VII)

We provide a self-contained proof that the Invariant Point Attention (IPA) attention weights are invariant under any global rigid transformation $g = (R, t) \in SE(3)$, following Jumper et al. (2021) with minor notational adjustments.

Setup. Let $T_i = (R_i, t_i) \in SE(3)$ be the rigid frame assigned to residue i, so that the global rotation R and translation t act by $T_i \mapsto g \circ T_i = (RR_i, Rt_i + t)$. For each attention head h, residue i has query points $\tilde{q}_{ih} \in \mathbb{R}^3$ expressed in the local frame of residue i, and key points $\tilde{k}_{jh} \in \mathbb{R}^3$ expressed in the local frame of residue j. Their global-frame positions are:

$$p_{ih} = R_i \tilde{q}_{ih} + t_i, \quad k_{jh} = R_j \tilde{k}_{jh} + t_j.$$

Step 1: Distance under g. After the global transformation $g = (R, t)$, the new global positions are $p'_{ih} = R(R_i \tilde{q}_{ih}) + (Rt_i + t)$ and similarly for k'_{jh}. The squared distance between the transformed query and key points is:

$$\| p'_{ih} - k'_{jh} \|^2 = \| R(R_i \tilde{q}_{ih} - R_j \tilde{k}_{jh}) \|^2 = \| R_i \tilde{q}_{ih} - R_j \tilde{k}_{jh} \|^2 = \| p_{ih} - k_{jh} \|^2$$

where the second equality uses the fact that $R \in SO(3)$ is an isometry ($\|Rv\| = \|v\|$ for all v). The translational term cancels because both p' and k' receive the same translation $Rt_i + t$ and $Rt_j + t$ respectively, whose difference is $R(t_i - t_j)$, which is absorbed into the rotation-invariant norm.

Step 2: Invariance of attention logits. The IPA attention logit between residues i and j is:

$$a_{ij} = (1/\sqrt{(d_h + \gamma m)}) \, [q_i^T k_j + (\gamma m/2) \, \Sigma_h \, \| p_{ih} - k_{jh} \|^2]$$

The first term $q_i^T k_j$ is a dot product between sequence-space features and is independent of the 3D frames entirely. The second term is a sum of squared distances in global frame, which Step 1 shows is invariant under g. Therefore a_{ij} is SE(3)-invariant, and so are the softmax weights $A_{ij} = \mathrm{softmax}_j(a_{ij})$. ■

B.2 Clebsch-Gordan Decomposition for $\ell \leq 2$

The following table gives the irreducible decomposition of the tensor product $V_{\ell_1}^1 \otimes V_{\ell_2}^2$ for angular momentum values $\ell_1, \ell_2 \le 2$, relevant to the equivariant message-passing architectures NequIP, MACE, and SEGNN. The triangle rule $|\ell_1 - \ell_2| \le \ell \le \ell_1 + \ell_2$ determines the output types.

$\ell_1 = 0, \ell_2 = 0$: $V^0 \otimes V^0 = V^0$ (scalar $\otimes$ scalar = scalar)

$\ell_1 = 0, \ell_2 = 1$: $V^0 \otimes V^1 = V^1$ (scalar $\otimes$ vector = vector)

$\ell_1 = 1, \ell_2 = 1$: $V^1 \otimes V^1 = V^0 \oplus V^1 \oplus V^2$ ($3 \otimes 3 = 1 \oplus 3 \oplus 5$)

$\ell_1 = 1, \ell_2 = 2$: $V^1 \otimes V^2 = V^1 \oplus V^2 \oplus V^3$ ($3 \otimes 5 = 3 \oplus 5 \oplus 7$)

$\ell_1 = 2, \ell_2 = 2$: $V^2 \otimes V^2 = V^0 \oplus V^1 \oplus V^2 \oplus V^3 \oplus V^4$ ($5 \otimes 5 = 1 \oplus 3 \oplus 5 \oplus 7 \oplus 9$)

The computational cost of the tensor product $V_{\ell_1}^1 \otimes V_{\ell_2}^2$ is $O((\ell_1 + \ell_2)^3)$ due to the Clebsch-Gordan coefficients $\langle \ell_1 m_1, \ell_2 m_2 \mid \ell m \rangle$ that must be computed and applied. For $\ell_max = 2$, this is at most $5^3 = 125$ operations per feature channel per edge, which MACE reduces using a body-order truncation that avoids high-order products.

B.3 Glossary of Geometric Deep Learning Terms

The following terms appear throughout Part VII with specialised meanings that may differ from their use in other subfields of mathematics.

Equivariance. A map $f: X \to Y$ between G-spaces is G-equivariant if $f(g \cdot x) = g \cdot f(x)$ for all $g \in G$. In molecular modelling, SE(3)-equivariance means that rotating/translating the input atoms produces a correspondingly rotated/translated output.

Frame. In the context of AlphaFold 2, a frame $T = (R, t) \in SE(3)$ is a rigid coordinate system attached to each residue, defined by the positions of the backbone atoms N, Cα, C via Gram-Schmidt orthogonalisation.

Irrep. An irreducible representation of a group G is a representation with no proper invariant subspaces. For SO(3), the irreps are labelled by $\ell \in \mathbb{N}_0$ and have dimension $2\ell + 1$. Every SO(3)-equivariant network decomposes its features into a direct sum of irreps.

LDDT. Local Distance Difference Test: the per-residue confidence metric output by AlphaFold 2 (pLDDT), measuring the fraction of predicted inter-residue distances within 0.5, 1, 2, and 4 Å of the true distance, averaged over neighbours within 15 Å.

MSA. Multiple Sequence Alignment: an alignment of homologous protein sequences that encodes evolutionary co-variation. Co-varying positions tend to be in 3D contact; the Evoformer exploits this signal via axial attention and triangular updates.

Pairformer. The simplified successor to the Evoformer in AlphaFold 3, which operates on the pair representation alone (no MSA dimension) using triangle attention and multiplicative updates.

Wigner D-matrix. The matrix representation $D_l(R)$ of a rotation $R \in SO(3)$ in the ℓ-th irrep. Entries are given by $D_l_mm'(\alpha, \beta, \gamma)$ where α, β, γ are the Euler angles of R. Wigner D-matrices satisfy the composition rule $D_l(R_1R_2) = D_l(R_1)D_l(R_2)$.

Appendix C: Technical Supplement for Part VIII — Agentic AI, Meta-Learning, and Continual Learning

This appendix provides extended derivations and definitions supporting Part VIII. All notation follows Appendix A.

C.1 POMDP Formalism: Extended Definitions and the Belief-State MDP

A Partially Observable Markov Decision Process (POMDP) is defined by the tuple (S, A, O, P, R, Ω, γ) introduced in Part VIII, Chapter 1. The key reduction to a belief-state MDP proceeds as follows.

Definition C.1 (Belief state). The belief state at time t is the posterior distribution $b_t \in \Delta(S)$ defined by $b_t(s) = P(S_t = s \mid o_1, a_1, \ldots, o_t)$. It is updated via Bayes' rule after each observation:

$$b'(s') = \Omega(o' \mid s', a) \, \Sigma_s P(s' \mid s, a) \, b(s) \; / \; P(o' \mid b, a)$$

This defines the belief update operator $\tau: \Delta(S) \times A \times O \to \Delta(S)$. The belief-state MDP has state space $\Delta(S)$ (a continuous simplex), action space A, and reward $\underline{R}(b, a) = \Sigma_s \, b(s) \, R(s, a)$. An optimal POMDP policy $\pi^*: \Delta(S) \to A$ maps belief states to actions and is equivalent to an optimal policy for the belief-state MDP.

Definition C.2 (Value function of a POMDP). The optimal value function $V^*: \Delta(S) \to \mathbb{R}$ satisfies the Bellman equation:

$$V^*(b) = \max_a \left[\underline{R}(b, a) + \gamma \, \Sigma_{o'} P(o' \mid b, a) \, V^*(\tau(b, a, o')) \right]$$

V^* is piecewise linear and convex in b (it is the upper envelope of a set of linear α-vectors). Exact POMDP solvers (PBVI, SARSOP) approximate V^* over a finite set of belief points. For LLM agents, the context window serves as a finite-dimensional approximation to b, with the attention mechanism implementing a soft version of the belief update operator τ.

C.2 EWC as Laplace Approximation: Complete Derivation

We derive the Elastic Weight Consolidation objective from the Bayesian perspective, filling in the steps sketched in Theorem 5.1 of Part VIII.

Starting point. After learning task $\mathcal{T}_1$, the posterior over parameters is $p(\theta \mid \mathcal{D}_1)$. When learning task $\mathcal{T}_2$, Bayes' rule gives:

$$\log p(\theta \mid \mathcal{D}_1, \mathcal{D}_2) = \log p(\mathcal{D}_2 \mid \theta) + \log p(\theta \mid \mathcal{D}_1) + \text{const}$$

Laplace approximation of $p(\theta \mid \mathcal{D}_1)$. At the MAP estimate $\theta_1^* = \arg\max p(\theta \mid \mathcal{D}_1)$, the second-order Taylor expansion gives:

$$\log p(\theta \mid \mathcal{D}_1) \approx \log p(\theta_1^* \mid \mathcal{D}_1) - (1/2)(\theta - \theta_1^*)^T H_1 (\theta - \theta_1^*)$$

where $H_1 = -\nabla^2_\theta \log p(\theta_1^* \mid \mathcal{D}_1)$ is the Hessian of the negative log-posterior at θ_1^*. Since $\log p(\theta \mid \mathcal{D}_1) = \log p(\mathcal{D}_1 \mid \theta) + \log p(\theta) - \text{const}$, and using the Fisher identity $F_1 \approx H_1$ (valid at a maximum likelihood point for a well-specified model), we obtain:

$$\log p(\theta \mid \mathcal{D}_1) \approx \text{const} - (1/2) \sum_i F_{1,ii} (\theta_i - \theta_{1,i}^*)^2$$

where the last step uses the diagonal approximation $F_1 \approx \text{diag}(F_{1,11}, \ldots, F_{1,pp})$. The MAP estimator for the joint posterior then minimises:

$$\mathcal{L}_{\text{EWC}}(\theta) = \mathcal{L}_{\{\mathcal{T}_2\}}(\theta) + (\lambda/2) \sum_i F_{1,ii} (\theta_i - \theta_{1,i}^*)^2$$

which is precisely the EWC objective. The diagonal Fisher $F_{1,ii} = \mathbb{E}_{p_{\text{data}}}[(\partial \log p_\theta(y \mid x) / \partial \theta_i)^2]$ at $\theta = \theta_1^*$ is computed on the task-1 training set as a sample average. ■

C.3 Topological Data Analysis: Key Definitions

This section provides precise definitions for the topological objects used in Part VIII, Chapter 6.

Definition C.3 (Simplicial complex). A simplicial complex K on a vertex set V is a collection of subsets (simplices) of V such that (i) every singleton $\{v\} \in K$, and (ii) if $\sigma \in K$ and $\tau \subseteq \sigma$, then $\tau \in K$. A k-simplex has $k+1$ vertices. The Vietoris-Rips complex $\text{VR}_\varepsilon(X)$ at scale ε is a simplicial complex on point cloud X where a subset $\sigma \subseteq X$ is a simplex iff $\text{diam}(\sigma) \leq \varepsilon$.

Definition C.4 (Homology group). For a simplicial complex K with coefficient ring $\mathbb{Z}/2\mathbb{Z}$, the k-th homology group $H_k(K) = \ker(\partial_k) / \text{im}(\partial_{k+1})$ where ∂_k is the boundary operator mapping k-chains to $(k-1)$-chains. The rank $\beta_k = \text{rank}(H_k(K))$ is the k-th Betti number: β_0 = connected components, β_1 = independent loops, β_2 = enclosed voids.

Definition C.5 (Persistence diagram). Given a filtration $K_0 \subseteq K_1 \subseteq \ldots \subseteq K_n$, a homological feature (class $\alpha \in H_k$) is born at $\varepsilon_b = \min\{\varepsilon_i : \alpha \in \text{im}(H_k(K_i) \to H_k(K_n))\}$ and dies at $\varepsilon_d = \min\{\varepsilon_j > \varepsilon_b : \alpha \in \text{im}(H_k(K_b) \to H_k(K_j)) = 0\}$. The persistence of α is $\text{pers}(\alpha) = \varepsilon_d - \varepsilon_b$. The persistence diagram D_k is the multiset of birth-death pairs $\{(\varepsilon_b, \varepsilon_d)\}$ for all k-dimensional homological features.

Remark C.1 (Bottleneck distance and stability). The bottleneck distance $d_B(D, D') = \inf_{\gamma} \sup_{p \in D} \|p - \gamma(p)\|_\infty$ between persistence diagrams D, D' satisfies the stability theorem: if $f, g: X \to \mathbb{R}$ with $\|f - g\|_\infty \leq \varepsilon$, then $d_B(D(f), D(g)) \leq \varepsilon$. This guarantees that small perturbations of the loss landscape or network weights produce small changes in the persistence diagram, validating topological training diagnostics.

C.4 GRPO Objective: Detailed Derivation from PPO

GRPO (Group Relative Policy Optimisation) modifies PPO by replacing the learned value function baseline with a within-group reward normalisation. Here we trace the derivation carefully.

PPO baseline. PPO minimises the surrogate loss $\mathscr{L}$_PPO = −𝔼[min(r(θ)Â, clip(r(θ), 1−ε, 1+ε)Â)], where r(θ) = π_θ(y|x) / π_{θ_old}(y|x) is the probability ratio and Â = (r − V_φ(x)) is the advantage estimated using a learned value function V_φ.

GRPO substitution. For a problem x and G sampled responses {y_1, ..., y_G} from π_{θ_old}, the group-relative advantage replaces V_φ(x) with the within-group mean reward:

$$\hat{A}_i = (r(x, y_i) - \text{mean}_j\, r(x, y_j)) \,/\, \text{std}_j\, r(x, y_j)$$

This is an unbiased estimator of the advantage: 𝔼_{y~π}[Â_i] = 0 by the centring. The variance of Â_i is O(1/G): larger group size G reduces variance as more within-group comparisons are made. The KL penalty β D_KL(π_θ ‖ π_ref) prevents the policy from drifting too far from a reference (pre-RLHF) model, playing the same role as the KL term in RLHF with PPO. The elimination of V_φ saves approximately one forward pass and one backward pass per training step (the critic network), halving memory usage in practice.

Bibliography for Part VII: Scientific AI and Geometric Deep Learning

The following references are cited in Part VII. Works are listed alphabetically by first author's surname. All arxiv preprints were accessed and verified as of publication date of this treatise.

Primary Architecture Papers

Abramson, J., Adler, J., Dunger, J., et al. (2024). Accurate structure prediction of biomolecular interactions with AlphaFold 3. Nature, 630, 493–500. doi:10.1038/s41586-024-07487-w. Introduces AlphaFold 3, extending structure prediction to DNA, RNA, and small molecules via a diffusion module operating on 3D atom coordinates.

Benegas, G., Bhodiwal, S., Ye, C., & Song, Y. S. (2024). AlphaGenome: Advancing genomic sequence and variant effect prediction. bioRxiv. Presents a unified multi-task model for regulatory genomics, predicting RNA-seq, ATAC-seq, ChIP-seq, and Hi-C from raw DNA sequence using multi-scale convolution and reverse complement equivariance.

Hayes, T., Rao, R., Akin, H., et al. (2024). Simulating 500 million years of evolution with a language model. Science, 385(6707). Introduces ESM-3, a multimodal protein language model jointly modelling sequence, structure, and function through unified masked language modelling.

Jumper, J., Evans, R., Pritzel, A., et al. (2021). Highly accurate protein structure prediction with AlphaFold. Nature, 596, 583–589. doi:10.1038/s41586-021-03819-2. The definitive paper on AlphaFold 2; introduces the Evoformer, Invariant Point Attention, FAPE loss, and the dual MSA/pair representation.

Equivariant Networks and Geometric Deep Learning

Batzner, S., Musaelian, A., Sun, L., et al. (2022). E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. Nature Communications, 13, 2453. Introduces NequIP, a highly data-efficient interatomic potential using tensor-product-based SE(3)-equivariant message passing.

Batatia, I., Kovács, D. P., Simm, G., Ortner, C., & Csányi, G. (2022). MACE: Higher order equivariant message passing neural networks for fast and accurate force fields. NeurIPS 2022. Introduces MACE, which uses body-order truncation to make high-ℓ tensor products computationally feasible.

Cohen, T. S. & Welling, M. (2016). Group equivariant convolutional networks. ICML 2016. The foundational paper establishing group-equivariant CNNs; proves that convolution is the unique linear equivariant operation for locally compact groups.

Cohen, T. S., Geiger, M., Köhler, J., & Welling, M. (2018). Spherical CNNs. ICLR 2018. Extends equivariant CNNs to the sphere S^2 using the spherical harmonics as the relevant irreducible representations of SO(3).

Klicpera, J., Groß, J., & Günnemann, S. (2020). Directional message passing for molecular graphs. ICLR 2020. Introduces DimeNet, which incorporates bond angles as directional features, strictly increasing the expressivity of distance-only GNNs.

Liao, Y. L. & Smidt, T. (2023). Equiformer: Equivariant graph attention transformer for 3D atomistic graphs. ICLR 2023. Brings the Transformer architecture to SE(3)-equivariant settings via equivariant dot-product attention.

Schütt, K., Kindermans, P.-J., Felix, H. E. S., Chmiela, S., Tkatchenko, A., & Müller, K.-R. (2017). SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. NeurIPS 2017. Introduces the first scalable distance-based GNN for molecular property prediction.

Schütt, K., Unke, O., & Gastegger, M. (2021). Equivariant message passing for the prediction of tensorial properties and molecular spectra. ICML 2021. Introduces PaiNN, which augments scalar features with equivariant vector features, enabling force prediction and chiral discrimination.

Satorras, V. G., Hoogeboom, E., & Welling, M. (2021). E(n) equivariant graph neural networks. ICML 2021. A computationally lightweight alternative to spherical-harmonic-based equivariant GNNs, achieving E(n)-equivariance without explicit irrep decompositions.

Thomas, N., Smidt, T., Kearnes, S., et al. (2018). Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds. arxiv:1802.08219. The first explicit use of spherical harmonics and Clebsch-Gordan products in equivariant neural networks for molecular data.

Group Theory and Representation Theory

Bronstein, M. M., Bruna, J., Cohen, T., & Veličković, P. (2021). Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. arxiv:2104.13478. The comprehensive mathematical treatment of geometric deep learning as a unification of existing architectures through the lens of symmetry and equivariance.

Fulton, W. & Harris, J. (1991). Representation Theory: A First Course. Springer Graduate Texts in Mathematics, vol. 129. The standard reference for the representation theory of Lie groups, including SO(3), SU(2), and their Clebsch-Gordan theory.

Weyl, H. (1946). The Classical Groups: Their Invariants and Representations (2nd ed.). Princeton University Press. The classical reference connecting group invariant theory to physics and geometry, underpinning the equivariance framework.

Multi-Task Learning and Gradient Methods

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., & Finn, C. (2020). Gradient surgery for multi-task learning. NeurIPS 2020. Introduces PCGrad, the gradient projection method for resolving conflicting gradients in multi-task settings, applied to regulatory genomics in AlphaGenome.

Bibliography for Part VIII: Agentic AI, Meta-Learning, and Continual Learning

The following references are cited in Part VIII. Works are listed alphabetically by first author's surname.

Agentic Systems and Multi-Agent Reinforcement Learning

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., & Zhou, D. (2023). What learning algorithm is in-context learning? Investigations with linear models. ICLR 2023. Proves that transformer in-context learning implements ridge regression via a gradient-descent-like mechanism in the attention layers.

Chaudhry, A., Ranzato, M. A., Rohrbach, M., & Elhoseiny, M. (2019). Efficient lifelong learning with A-GEM. ICLR 2019. Introduces A-GEM, a computationally efficient approximation to GEM that projects the gradient onto a single averaged past-task constraint.

Lasry, J.-M. & Lions, P.-L. (2007). Mean field games. Japanese Journal of Mathematics, 2(1), 229–260. The foundational paper establishing mean-field game theory and the forward-backward equation system for the N→∞ limit.

Lopez-Paz, D. & Ranzato, M. A. (2017). Gradient episodic memory for continual learning. NeurIPS 2017. Introduces GEM, which enforces non-negative inner products between the current gradient and stored past-task gradients as an exact non-forgetting constraint.

Rashid, T., Samvelyan, M., Schroeder de Witt, C., Farquhar, G., Foerster, J., & Whiteson, S. (2018). QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. ICML 2018. Introduces the QMIX mixing network and the Individual-Global-Max (IGM) condition for cooperative MARL.

Yao, S., Zhao, J., Yu, D., et al. (2022). ReAct: Synergizing reasoning and acting in language models. ICLR 2023. Introduces the ReAct framework for LLM agents, interleaving chain-of-thought reasoning with tool-call actions within the context window.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., & Wu, Y. (2022). The surprising effectiveness of PPO in cooperative multi-agent games. NeurIPS 2022. Demonstrates that MAPPO with centralised critics is competitive with specialised MARL algorithms on StarCraft and other benchmarks.

Inference-Time Compute and Reinforcement Learning from Human Feedback

Coulom, R. (2006). Efficient selectivity and backup operators in Monte-Carlo tree search. Computers and Games, LNCS 4630. The paper introducing MCTS with the UCB1 selection rule; foundational for AlphaGo and subsequent work on MCTS for LLMs.

Lightman, H., Kosaraju, V., Burda, Y., et al. (2023). Let's verify step by step. arxiv:2305.20050. Introduces Process Reward Models trained on step-by-step human annotations,

demonstrating their superiority over outcome-only reward models for mathematical reasoning.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization algorithms. arxiv:1707.06347. The PPO paper; introduces the clipped surrogate objective and the KL-penalty variant, both used in GRPO.

Shao, Z., Wang, P., Zhu, Q., et al. (2024). DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arxiv:2402.03300. Introduces GRPO (Group Relative Policy Optimisation) and demonstrates its effectiveness for mathematical reasoning without a critic network.

Snell, C., Lee, J., Xu, K., & Kumar, A. (2024). Scaling LLM test-time compute optimally. arxiv:2408.03314. A systematic study of test-time compute scaling strategies (Best-of-N, beam search, PRM-guided MCTS), establishing the optimal allocation of inference-time compute.

Video Generation and Diffusion Models

Brooks, T., Peebles, B., Holmes, C., et al. (2024). Video generation models as world simulators. OpenAI Technical Report. Introduces Sora: a video generation model based on DiT with spatiotemporal patch tokenisation and 3D RoPE, trained to produce physically coherent videos of arbitrary resolution and duration.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., & Fleet, D. J. (2022). Video Diffusion Models. NeurIPS 2022. The foundational paper on diffusion models for video generation; introduces factored spatial and temporal attention to reduce the $O((THW)^2)$ computational cost of full spatiotemporal attention.

Yang, Z., Teng, J., Zheng, W., et al. (2024). CogVideoX: Text-to-video diffusion models with an expert transformer. arxiv:2408.06072. Demonstrates high-quality text-to-video synthesis using 3D full attention with flow matching in approximately 50 sampling steps.

Meta-Learning

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. ICML 2017. Introduces MAML; proves that a single gradient step on a small support set, followed by a meta-gradient through this step, produces fast task adaptation across diverse problem domains.

Nichol, A., Achiam, J., & Schulman, J. (2018). On first-order meta-learning algorithms. arxiv:1803.02999. Introduces Reptile and FOMAML; proves that Reptile implicitly maximises within-task gradient alignment, providing a first-order justification for MAML-like meta-learning.

Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. NeurIPS 2017. Introduces ProtoNets, where class prototypes are the means of support-set embeddings and classification uses nearest-prototype in Euclidean space.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., & Wierstra, D. (2016). Matching networks for one shot learning. NeurIPS 2016. The seminal few-shot learning paper

introducing attention-based matching over a support set, predating and motivating subsequent Transformer-based approaches.

Continual Learning

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., et al. (2017). Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13), 3521–3526. Introduces Elastic Weight Consolidation (EWC), using the Fisher information matrix diagonal as parameter importance weights in an L2-regularisation term that penalises forgetting.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., et al. (2016). Progressive neural networks. arxiv:1606.04671. Introduces Progressive Neural Networks, which add a new column of parameters per task with lateral connections to previous frozen columns, guaranteeing zero forgetting by construction.

Zenke, F., Poole, B., & Ganguli, S. (2017). Continual learning through synaptic intelligence. ICML 2017. Introduces Synaptic Intelligence (SI), an online variant of EWC that accumulates parameter importance weights during training based on each parameter's contribution to the loss decrease.

Topological Analysis of Learning

Carlsson, G. (2009). Topology and data. Bulletin of the American Mathematical Society, 46(2), 255–308. The foundational survey of persistent homology and topological data analysis; introduces the persistence diagram and the stability theorem.

Cohen-Steiner, D., Edelsbrunner, H., & Harer, J. (2007). Stability of persistence diagrams. Discrete and Computational Geometry, 37(1), 103–120. Proves the bottleneck stability theorem (Theorem 6.1 of Part VIII), establishing that persistence diagrams vary continuously with the underlying function.

Frankle, J. & Carlin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. ICLR 2019. Empirically identifies sparse subnetworks ("winning tickets") that match the performance of dense networks when trained in isolation, motivating the study of loss landscape geometry.

Nanda, N., Chan, L., Lieberum, T., Smith, J., & Steinhardt, J. (2023). Progress measures for grokking via mechanistic interpretability. ICLR 2023. Provides the mechanistic account of grokking for modular arithmetic; identifies the Fourier-circular representation and the phase transition in H_1 persistent homology as the mechanism of delayed generalisation.

Naitzat, G., Zhitnikov, A., & Lim, L.-H. (2020). Topology of deep neural networks. Journal of Machine Learning Research, 21(184), 1–40. Analyses the topological transformation of data representations across layers, showing that deep networks progressively reduce the Betti numbers of class boundaries.